Supplementary information of

A unified framework to understand and control dynamics in Boolean biological networks

Antonio Bensussen[1]*, Elena R. Álvarez-Buylla[2], Juan Carlos Martínez-García[1]*

[1]Departamento de Control Automático, Cinvestav-IPN, Ciudad de México, México.

[2]Instituto de Ecología, Universidad Nacional Autónoma de México, Ciudad de México, México.

*** Correspondence to:** Antonio Bensussen (antonio.bensussen@cinvestav.mx), Juan Carlos Martínez-García (juancarlos.martinez@cinvestav.mx).

Functional representation of Boolean networks

The computational modeling of real biological processes may require a large number of parameters that are difficult to estimate experimentally (1,2). In this respect, Boolean models constitute a powerful tool for overcoming this technical problem, because they allow the direct use of process logic to create mechanistic computational representations (3). Consequently, it is easy to use real experimental data to support and validate Boolean models (4,5). Since its first application to biology, proposed by Stuart Kauffman, to the present day, each molecule or gene represented in a Boolean model act as a node interconnected with other nodes through its biological interactions (3). In this way, the interactions of node x_1 with the set of nodes $\{x_1, x_2, \dots, x_n\}$ can be abstractly represented as:

$$x_1(t+1) = f_1\big(x_1(t), x_2(t), \dots, x_n(t)\big)$$

Where, $x_1(t+1)$ represents the future state of node x_1, $x_i(t)$ represents the current state of x_i, and f_1 is a logical function that connects x_1 with all the molecules or genes involved in the biological process being studied, i.e., $\{x_1, x_2, \dots, x_n\}$ (3). Consequently, the Boolean network that represents a biological process composed of the set of genes and molecules $\{x_1, x_2, \dots, x_n\}$ can be written as:

$$\begin{aligned} x_1(t+1) &= f_1\big(x_1(t), x_2(t), \dots, x_n(t)\big) \\ x_2(t+1) &= f_2\big(x_1(t), x_2(t), \dots, x_n(t)\big) \\ &\vdots \\ x_n(t+1) &= f_n\big(x_1(t), x_2(t), \dots, x_n(t)\big) \end{aligned} \qquad \text{(E. 1)}$$

Starting from this classic definition of a Boolean network, we might simplify the notation as follows:

$$\begin{aligned} x_1^+ &= f_1(x_1, x_2, \dots, x_n) \\ x_2^+ &= f_2(x_1, x_2, \dots, x_n) \\ &\vdots \\ x_n^+ &= f_n(x_1, x_2, \dots, x_n) \end{aligned} \qquad \text{(E. 2)}$$

Where $x_i^+ = x_i(t+1)$, and $x_i = x_i(t)$, $i = 1,2, \dots, n$. Observe that all possible configurations of this network are contained in the state space (Ω) which has indexed elements from 0 to $2^n - 1$, and its cardinality is given by $|\Omega| = 2^n$, where n is the number of nodes of the network. After making this modification, note that we may write in a Boolean network F that updates synchronously; that is, all nodes are updated at the same time, as follows:

$$F_{2^n-1}(x_1^+, x_2^+, \dots, x_n^+) = (f_1, f_2, \dots, f_n) \qquad \text{(E. 3)}$$

Where $f_i = f_i(x_1, x_2, \dots, x_n)$. From this redefinition of a Boolean network, we shall define the update of one or more nodes individually as indicated below.

Definition 1 (Update function): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes. We shall say that F_k is an update function of F if the state of a set of k changes due to the action of

one or more logic functions, where $0 < k < 2^n - 1$. However, if no node is updated, the following will occur:

Definition 2 (Boolean function of conservation): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes. We shall say that F_0 is a Boolean function of conservation, because if no node is updated, then there is linear continuity between the initial state and the future state of the network. That is: $F_0(x_1, x_2, \dots, x_n) = (x_1, x_2, \dots, x_n)$.

Example 1: To integrate these concepts, consider the Boolean network given by:

$$\begin{cases} x_1^+ = x_1 x_2 \\ x_2^+ = x_1 + x_2 \end{cases} \quad \text{(E. 4)}$$

From this network, we can write the synchronous update function as:

$$F_3(x_1^+, x_2^+) = (x_1 x_2, x_1 + x_2)$$

Similarly, the asynchronous update functions would correspond to:

$$F_1(x_1^+, x_2) = (x_1 x_2, x_2)$$

$$F_2(x_2, x_2^+) = (x_1, x_1 + x_2)$$

Finally, the Boolean conservation function is given by:

$$F_0(x_1, x_2) = (x_1, x_2)$$

Therefore, any update of a Boolean network can be represented by an analytic function.

Functional representation of update schemes

Previously, we showed that the synchronous update scheme is equivalent to the composition of the synchronous update function, which corresponds to a particular case of a Markov chain (16). We also showed that the asynchronous scheme and all its dynamic features originate from the composition of update functions (16). In view of the above, we shall affirm that:

Definition 3 (Functional form of an update scheme): Let F be a Boolean network of n nodes. We say that the synchronous update scheme is obtained by iteratively composing the function $F_{2^n-1}(x_1^+, x_2^+, \dots, x_n^+)$. Any asynchronous or semi-asynchronous scheme is obtained by composing with the update functions defined by the scheme. The use of the Boolean conservation function implies a time delay.

Remark 1: There are 2^n possible update functions for each Boolean network of n nodes.

Example 2: To apply this concept, observe the following network:

$$\begin{cases} x_1^+ = x_1 + x_3 \\ x_2^+ = x_1 x_2 \\ x_3^+ = \bar{x}_2 \end{cases} \tag{E.5}$$

This network has a synchronous update function given by:

$$F_7(x_1^+, x_2^+, x_3^+) = (x_1 + x_3, x_1 x_2, \bar{x}_2)$$

Similarly, its asynchronous update functions are:

$$F_1(x_1, x_2, x_3^+) = (x_1, x_2, \bar{x}_2)$$

$$F_2(x_1, x_2^+, x_3) = (x_1, x_1 x_2, x_3)$$

$$F_3(x_1, x_2^+, x_3^+) = (x_1, x_1 x_2, \bar{x}_2)$$

$$F_4(x_1^+, x_2, x_3) = (x_1 + x_3, x_2, x_3)$$

$$F_5(x_1^+, x_2, x_3^+) = (x_1 + x_3, x_2, \bar{x}_2)$$

$$F_6(x_1^+, x_2^+, x_3) = (x_1 + x_3, x_1 x_2, x_3)$$

And its conservation function is:

$$F_0(x_1, x_2, x_3) = (x_1, x_2, x_3)$$

Fig. S1 shows the evaluation of all the update functions shown here. Now, let's suppose that we want to create two update schemes. The first one will be fully asynchronous, updating node x_1 first, then node x_2, and finally node x_3. The second one will be semi-asynchronous, updating node x_2 first, then x_1, and finally nodes x_2 and x_3 simultaneously. To do this, we simply need to take the update functions corresponding to each instruction in the scheme and combine them. Fig. S2A illustrates how to form the proposed pure asynchronous scheme in this example, while Fig. S2B shows how to form the corresponding semi-asynchronous scheme. It is noteworthy to say that this approach effectively generates the computational results of implementing the proposed update schemes in Visual Studio 2022 (Figs. S2C and S2D). These results demonstrate that it is possible to generate any update scheme from composition of update functions.

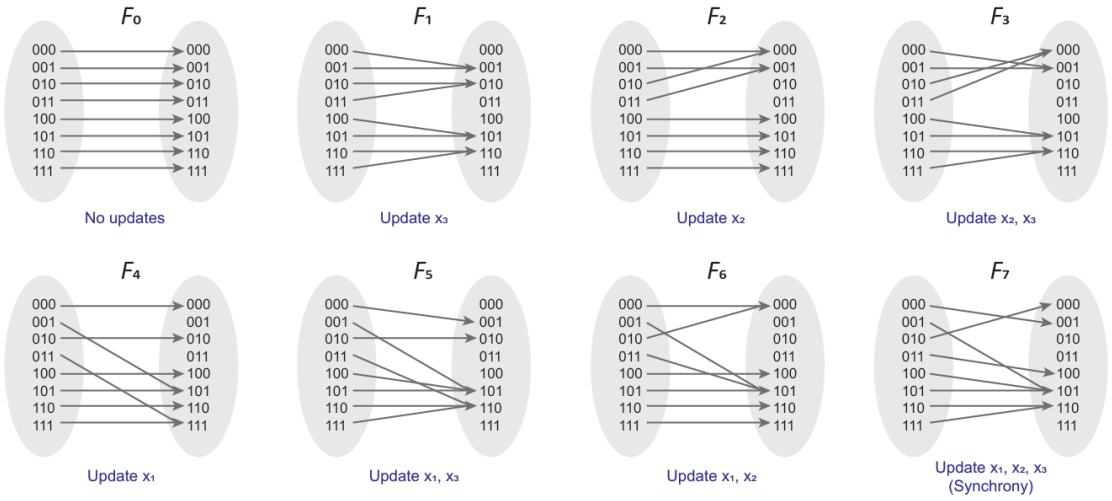

Figure S1. **Update functions derived from a Boolean network**. This figure shows the maps of the update functions derived from the network (E. 5). Each map is generated by evaluating each function with all configurations of the state space.

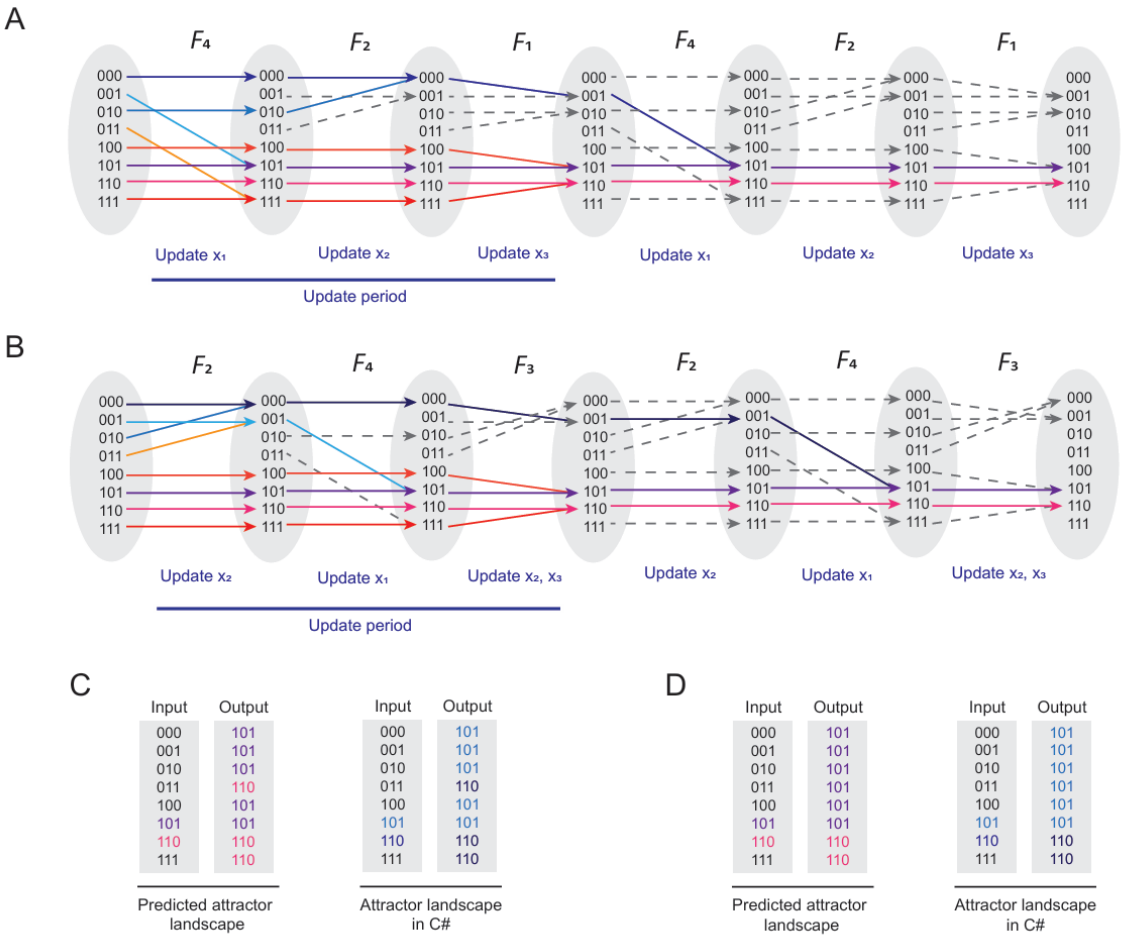

Figure S2. Analytical construction of update schemes. This figure shows how, using the update functions, it is possible to create any update scheme. (A) This panel shows the paths of the synchronous update scheme for the network (E. 5), where node x_1 is updated first, then node x_2, and finally node x_3. (B) This panel shows the paths of the semi-asynchronous update scheme, where x_2 is updated first, then x_1, and finally x_2 and x_3 are updated simultaneously. Panels (C) and (D) show a comparison of the predictions of the composition of the update functions (left) versus the results obtained by implementing the update schemes in C#.

Asynchronous and semi-asynchronous update schemes can be approximated by a synchronous update scheme

It is well known that asynchronous and semi-asynchronous update schemes are periodic (11,15), which implies that the order of the update functions corresponding to the first update period is repeated, as shown in Figs. S2A and S2B. Thus, if we consider only the functions corresponding to the first update period (Figs. S2A and S2B), we might compose them and generate a single update function. To recover the asynchronous and semi-asynchronous schemes, it is sufficient to repeat the update function obtained for the first update period. We previously demonstrated that the synchronous update scheme is recovered by composing the same update function to infinity (16), which also occurs with asynchronous and semi-asynchronous schemes when they are recovered from the update function of their period. This implies that asynchronous and semi-asynchronous schemes can be treated as synchronous schemes as long as their update period is known.

Example 3: Suppose we want to study the following Boolean network under an asynchronous update scheme in which x_1 is updated first, then x_2, and finally x_3.

$$\begin{cases} x_1^+ = x_2 \\ x_2^+ = x_3 \\ x_3^+ = x_1 + x_3 \end{cases} \tag{E.6}$$

From this network, the following update functions involved in this scheme are (Fig. S3A):

$$F_1(x_1^+, x_2, x_3) = (x_2, x_2, x_3)$$

$$F_2(x_1, x_2^+, x_3) = (x_1, x_3, x_3)$$

$$F_3(x_1, x_2, x_3^+) = (x_1, x_2, x_1 + x_3)$$

Now, to obtain the period function (F_T), we need to compose the update functions according to the order set out in the asynchronous scheme of the example, i.e.:

$$F_T(x_1, x_2, x_3) = F_3\left(F_2\big(F_1(x_1, x_2, x_3)\big)\right)$$

Which results in (Fig. S3B):

$$F_T(x_1, x_2, x_3) = (x_2, x_3, x_2 + x_3)$$

Note that the behavior of the originally proposed asynchronous scheme (Fig. S3C) and the behavior of the synchronous scheme derived from the composite function F_T (Fig. S3D) fully match with their computational implementation in Visual Studio 2022 (Figs. S3E and S3F), proving that they are equivalent. Interestingly, the F_T trajectories are similar to the asynchronous paths, although faster, as it can be seen in Figs. S3C and S3D. Therefore, it is possible to approximate any the asynchronous update schemes by using a function that is synchronously updated. Note that trajectory means the path (or orbit) of a point under an iterative function is the sequence of values generated by repeatedly applying the same function, using the result of each step as the input for the next.

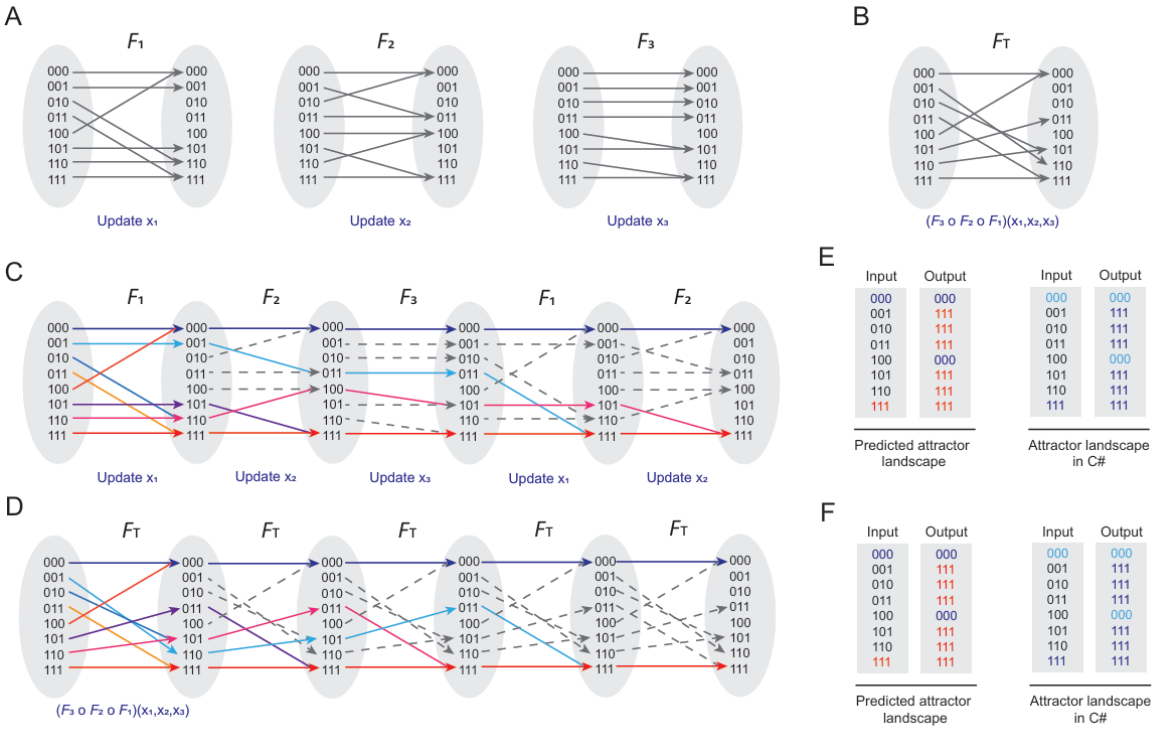

Figure S3. Approximating an asynchronous scheme with a synchronous scheme. This figure shows how an asynchronous scheme can be approximated by a synchronous update scheme. (A) This panel shows the maps of the update functions for the network (E. 6). (B) This panel shows the composition map of the update functions that constitute the asynchronous update scheme, in which x_1 is updated first, then x_2, and finally x_3. (C) This panel shows the evolution of the paths of the asynchronous scheme. (D) This panel shows the evolution of the paths of the image of the synchronous scheme generated from the function F_T. € Comparison between the attractor landscape predicted by the image paths for the asynchronous update scheme and its computational implementation in C#. (F) Comparison between the attractor landscape predicted by the image paths for the synchronous update scheme and its computational implementation in C#.

Canonical form of asynchronous and semi-asynchronous update schemes

In a previous study, we demonstrated the existence of the canonical form of Boolean networks; that is, a single network that possesses the minimum necessary and sufficient conditions to generate, in a single time step, the entire attractor landscape of any synchronously updated Boolean network (16). In that study, we also showed that the canonical form of a Boolean network allows to obtain the reachability conditions of any attractor and constituted an effective tool for solving large networks (16). All this mathematical development was limited exclusively to the synchronous update scheme, because we believed it was not possible to extrapolate it to other update schemes (16). However, the results of the previous section show that, by generating a function equivalent

to the update period of an asynchronous or semi-asynchronous scheme, we can treat this function as if it were a synchronous update scheme. This opens up the possibility of extrapolating the entire theoretical framework that we previously developed for the synchronous scheme to other update schemes, such as the asynchronous or semi-asynchronous schemes.

Example 4: To show how to develop the workflow for solving networks using the canonical form, let's consider the following Boolean network:

$$\begin{cases} x_1^+ = x_1 \\ x_2^+ = x_1 \\ x_3^+ = x_2 \\ x_4^+ = x_4 \\ x_5^+ = x_4 \\ x_6^+ = x_5 \\ x_7^+ = x_6 \\ x_8^+ = x_7 \\ x_9^+ = x_4 + x_8 \end{cases} \quad \text{(E. 7)}$$

Now, suppose we want to study this network with an asynchronous update scheme where x_1 is updated first, then x_2, and so on until x_9. To solve this network using the canonical form method developed by us in (16), we first obtain the update period function (Fig. S4A), then fragment the function into modules of arbitrary size (Fig. S4B). Next, we find the canonical form for each module (Fig. S4C), and from this, we find the local attractors for each module (Fig. S4D). These local attractors are then combined to form a list of semi-attractors (Fig. S4E), which must be evaluated only once on the original Boolean network. If the same configurations are obtained as a result, then they correspond to true attractors (Fig. S4E) (16). Therefore, it is possible to calculate the canonical form for any asynchronous and semi-asynchronous update schemes as long as their period is known.

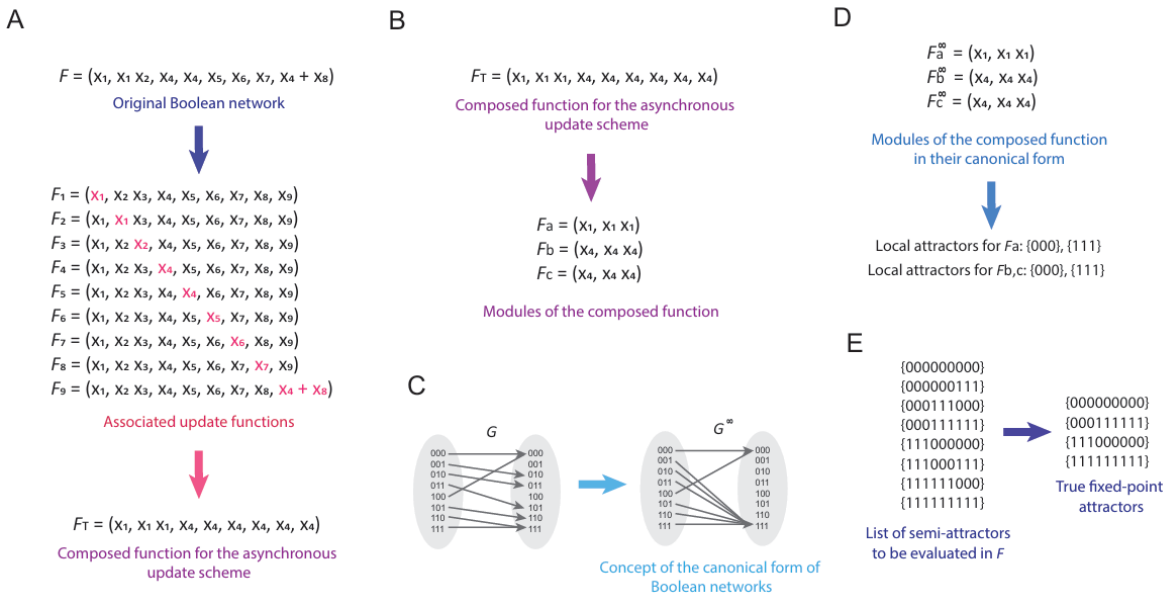

Figure S4. Workflow for finding fixed-point attractors in large scale networks. Panel (A) shows the update functions obtained from the original definition of the Boolean network F. Subsequently, by composing the update functions according to the system's update order, F_T is obtained, a synchronous function that summarizes the entire asynchronous scheme. Panel (B) shows how to fragment the F_T network into modules of arbitrary size; in this case, the modules have three nodes. (C) After obtaining the modules F_a, F_b, and F_c, the canonical form of each module must be obtained, as described in (16). This procedure involves collapsing all paths directly to the equilibrium point, which they will reach at infinity, as shown in the diagram in panel (C). Next, the local attractors for each module in canonical form must be found, as shown in panel (D). Finally, a list of semi-attractors, that is, the combination of the union of the attractors of each module, must be formed and evaluated only once in the original function F, as shown in panel (E). The result corresponds to the attractors of the F_T network.

Fixed-point attractors of the synchronous update scheme are global attractors

One of the main implications of the results presented so far is that all update schemes share a common origin, and it is possible to understand their dynamic characteristics. This includes predicting their attractors and associated attraction basins. Regarding attractors, Carlos Gershenson was the first to conclude, through computational experiments, that fixed-point attractors are independent of the update scheme used to study a Boolean network (15). Numerous subsequent works empirically confirmed this assertion (11,19–22). However, there is still no formal proof that demonstrates this fact or defines its context of validity. Therefore, we present below the formal proof of this claim, along with two auxiliary lemmas.

Lemma 1: Let $H: \mathbb{B}^n \to \mathbb{B}^n$ be a synchronously updated Boolean network. If $a \in \mathbb{B}^n$ is a fixed point of H, it will also be a fixed point for any composition of H with itself.

Proof: If $a \in \mathbb{B}^n$ is a fixed-point attractor of H, it implies that $H(a) = a$, then:

$$H\Big(H\Big(H\big(H(a)\big)\Big)\Big) = H\Big(H\big(H(a)\big)\Big) = H\big(H(a)\big) = H(a) = a$$

Therefore: $H^{(n)}(a) = a, \quad \forall n > 0$ ■

The intuition behind this demonstration can be observed in Fig. S5A.

Lemma 2: Let $H: \mathbb{B}^n \to \mathbb{B}^n$ and $G: \mathbb{B}^n \to \mathbb{B}^n$ be two Boolean networks that are synchronously updated. If $a \in \mathbb{B}^n$ is a fixed-point for both H and G, it will also be a fixed-point for any composition of H and G.

Proof: If $a \in \mathbb{B}^n$ is a fixed-point attractor of H, it implies that $H(a) = a$ and $G(a) = a$, then:

$$H\big(G(a)\big) = H(a) = a$$

As well as:

$$G\big(H(a)\big) = G(a) = a$$

Therefore: $(G \circ H)(a) = (H \circ G)(a) = a$ ■

The intuition behind this proof can be observed in Fig. S5B. Now, we will use these lemmas for the final proof.

Theorem 1 (Conservation of fixed-points): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes that is synchronously updated. If $a \in \mathbb{B}^n$ is a fixed-point attractor for F, then a is a fixed-point attractor for any update scheme derived from F, regardless of whether it is asynchronous or semi-asynchronous.

Proof: Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes. If F is updated synchronously, then:

$$F_{2^n-1}(x_1^+, x_2^+, \dots, x_n^+) = \big(f_1(x_1, x_2, \dots, x_n), f_2(x_1, x_2, \dots, x_n), \dots, f_n(x_1, x_2, \dots, x_n)\big)$$

Also, this network produces the following update functions:

$$F_1(x_1^+, x_2, \dots, x_n) = (f_1(x_1, x_2, \dots, x_n), x_2, \dots, x_n)$$

$$F_2(x_1, x_2^+, \dots, x_n) = (x_1, f_2(x_1, x_2, \dots, x_n), \dots, x_n)$$

$$F_3(x_1^+, x_2^+, \dots, x_n) = (f_1(x_1, x_2, \dots, x_n), f_2(x_1, x_2, \dots, x_n), \dots, x_n)$$

And so on for the rest of update functions originated from F. Also, this network has its conservation function given by:

$$F_0(x_1, x_2, \dots, x_n) = (x_1, x_2, \dots, x_n)$$

If $a \in \mathbb{B}^n$ is a fixed-point attractor for F_{2^n-1}, it implies that:

$$F_{2^n-1}(x_1^+, x_2^+, \dots, x_n^+)|_{(a_1, a_2, \dots, a_n)} = \left(f_1(a_1, a_2, \dots, a_n), f_2(a_1, a_2, \dots, a_n), \dots, f_n(a_1, a_2, \dots, a_n)\right) = (a_1, a_2, \dots, a_n)$$

Observe that:

$$F_1(x_1^+, x_2, \dots, x_n)|_{(a_1, a_2, \dots, a_n)} = (f_1(a_1, a_2, \dots, a_n), a_2, \dots, a_n) = (a_1, a_2, \dots, a_n)$$

$$F_2(x_1, x_2^+, \dots, x_n)|_{(a_1, a_2, \dots, a_n)} = (a_1, f_2(a_1, a_2, \dots, a_n), \dots, a_n) = (a_1, a_2, \dots, a_n)$$

$$F_3(x_1^+, x_2^+, \dots, x_n)|_{(a_1, a_2, \dots, a_n)} = (f_1(a_1, a_2, \dots, a_n), f_2(a_1, a_2, \dots, a_n), \dots, a_n) = (a_1, a_2, \dots, a_n)$$

And so on for the rest of update functions, including the conservation function, i.e.,

$$F_0(x_1, x_2, \dots, x_n)|_{(a_1, a_2, \dots, a_n)} = (a_1, a_2, \dots, a_n)$$

By Lemma 2, we already know that if $f(a) = a$ and $g(a) = a$, then $(f \circ g)(a) = (g \circ f)(a) = a$. Therefore, we might conclude that any composition of the update functions, which generates all update schemes derived from F, will maintain $(a_1, a_2, \dots, a_n)$ as a fixed-point attractor ■

Corollary 1: It does not matter whether the composition of the update functions is performed deterministically or stochastically, as long as $a \in \mathbb{B}^n$ is a fixed-point attractor of the synchronous implementation of $F: \mathbb{B}^n \to \mathbb{B}^n$, it will be so for any update scheme (Fig. S5C).

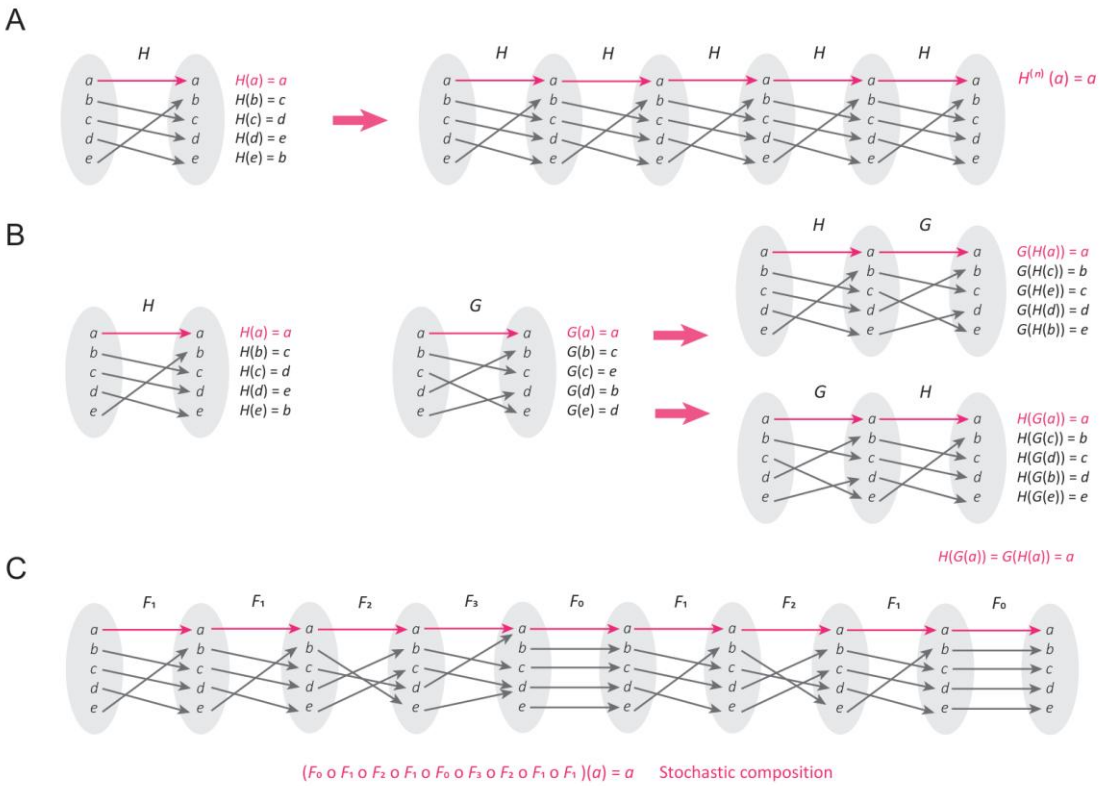

Figure S5. Conservation of fixed-point attractors of the synchronous update scheme. Panel (A) graphically illustrates the concept of a fixed-point attractor. Each time H is applied, the attractor path, a, remains constant. Panel (B) graphically shows that if a state is a fixed-point attractor for two different networks, it will also be a fixed-point attractor for the composition of those functions, regardless of the order. The main implication of this is that non-synchronous update schemes originate from the composition of multiple update functions. Therefore, if all the update functions have a state as a fixed-point attractor, that state will still appear regardless of the order in which the update functions are composed. Finally, panel (C) shows that even if the composition is stochastic, if a state is a fixed-point attractor for all the update functions, it will remain so despite the stochasticity.

Remark 2: Stochastic update schemes cannot be expressed as a unique function because they do not have a period.

Example 5: To illustrate these concepts, let's consider the following Boolean network:

$$\begin{cases} x_1^+ = x_2 + \bar{x}_2 \\ x_2^+ = \bar{x}_1 x_1 \end{cases} \quad \text{(E. 8)}$$

Note that this function has the following associated functions (Fig. S6A):

$$F_3(x_1^+, x_2^+) = (x_2 + \bar{x}_2, \bar{x}_1 x_1)$$

$$F_2(x_1^+, x_2) = (x_2 + \bar{x}_2, x_2)$$

$$F_1(x_1, x_2^+) = (x_1, \bar{x}_1 x_1)$$

$$F_0(x_1, x_2) = (x_1, x_2)$$

Note that the function F_3, corresponding to the synchronous update (Fig. S6B), produces a single fixed-point attractor at $\{10\}$, while the asynchronous update scheme, resulting from first updating node x_2, pausing, and then updating it again, generates fixed-point attractors $\{00\}$ and $\{10\}$ (Figs. S6C). Observe that when the update of node x_1 is included or when both nodes are updated simultaneously, the attractor $\{00\}$ disappears. On the other hand, if we use a stochastic update scheme (Fig. S6D), it can be observed that only the attractors of the synchronous scheme are maintained. These observations are explained by the fact that in Theorem 1 we proved that the only fixed-point attractors for the synchronous update scheme are conserved across all update schemes. Therefore, all the fixed-point attractors of the synchronous scheme are global attractors.

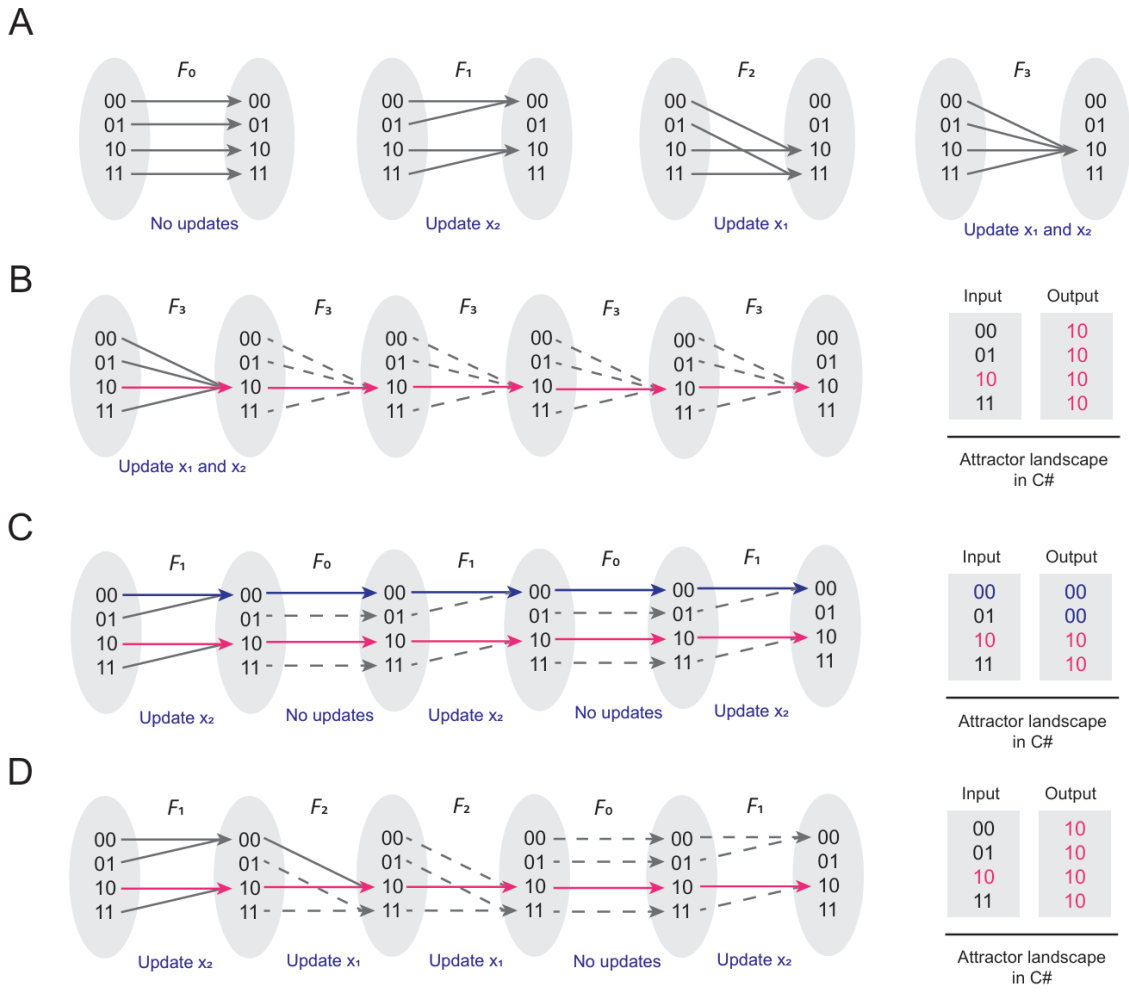

Figure S6. Emergence of "synchronous-sensitive" attractors. Panel (A) shows all the update functions of the network (E. 8). Panel (B) shows the evolution of the paths in a synchronous update scheme. Note that the diagram presented in this panel generates exactly the same attractor landscape as the synchronous implementation of the network (E. 8) in C#. Panel (C) shows the emergence of the synchronized attractor $\{00\}$, as a result of the composition of functions F_1 and F_0, for which this state is a true attractor, as verified in C#. Finally, panel (D) shows the evolution of the paths resulting from the stochastic composition of the update functions. Note that this composition generates exactly the same attractor landscape as the synchronous scheme, a prediction that was validated in C#.

Synchrony-sensitive attractors are local fixed-points

In apparent contradiction to Gershenson's work, Noual and Sené, for their part, empirically observed that asynchronous schemes can generate attractors that are "synchrony-sensitive", that is, fixed-point attractors that appear under certain asynchronous schemes but disappear under synchronous update (23). This observation is explained by Theorem 1, since we showed that only the fixed-point attractors of the synchronous scheme are conserved in all update schemes. However, there may be update functions that, when combined, generate fixed-point attractors not included in the synchronous update scheme. These fixed-point attractors correspond to "synchrony-sensitive attractors," since they disappear when the

update conditions change or the synchronous scheme is used. As evidence of this, note in Example 5 how the attractor $\{00\}$, resulting from an asynchronous update scheme, disappears when the update scheme changes, while the attractor of the synchronous scheme $\{10\}$ remains (Fig. S6C). Consequently, Noual's observations do not contradict Gershenson's work, since, in reality, "synchrony-sensitive" attractors are not globally valid. Thus, “synchrony-sensitive” fixed-points are indeed local attractors.

Synchrony-sensitive attractors can be rationally designed

Previously, we discovered that by choosing the update order for asynchronous update scheme, it was possible to modify the size and constitution of the basins of attraction, since the composition of the update functions could bias the system's trajectories in the desired direction (16). Interestingly, we observed that such principle can be used to create “synchrony-sensitive” fixed-point attractors (Fig. S6C). The key lies in leaving some nodes without update or by using update functions that do not affect certain trajectories. This can be achieved by mapping the update functions and searching in the map for unchanging trajectories, distinct from the fixed-point attractors of the synchronous update scheme. Consequently, the more nodes that remain unchanged, the greater the number of synchrony-sensitive attractors that will emerge.

Example 6: To illustrate these principles, consider the following Boolean network:

$$\begin{cases} x_1^+ = \bar{x}_1 x_2 \bar{x}_3 + x_1 \\ x_2^+ = x_1 x_3 \\ x_3^+ = x_1 x_3 \end{cases} \tag{E. 9}$$

This network has the following update functions:

$$F_7(x_1^+, x_2^+, x_3^+) = (\bar{x}_1 x_2 \bar{x}_3 + x_1, x_1 x_3, x_1 x_3)$$

$$F_6(x_1^+, x_2^+, x_3) = (\bar{x}_1 x_2 \bar{x}_3 + x_1, x_1 x_3, x_3)$$

$$F_5(x_1^+, x_2, x_3^+) = (\bar{x}_1 x_2 \bar{x}_3 + x_1, x_2, x_1 x_3)$$

$$F_4(x_1^+, x_2, x_3) = (\bar{x}_1 x_2 \bar{x}_3 + x_1, x_2, x_3)$$

$$F_3(x_1, x_2^+, x_3^+) = (x_1, x_1 x_3, x_1 x_3)$$

$$F_2(x_1, x_2^+, x_3) = (x_1, x_1 x_3, x_3)$$

$$F_1(x_1, x_2, x_3^+) = (x_1, x_2, x_1 x_3)$$

$$F_0(x_1, x_2, x_3) = (x_1, x_2, x_3)$$

Note that the maps associated with each function are shown in Fig. S7A. As it can be seen in this figure, depending on the node being updated, certain paths appear that remain constant. In this regard, it is noteworthy to say that for the update functions F_4 and F_2, as well as for the conservation function F_0, the state $\{001\}$ is a fixed-point attractor (Fig. S7B). For this

state to emerge as a "synchrony-sensitive" attractor, it is sufficient to propose an update scheme that does not modify it, such as updating x_1, waiting one time step, updating x_2, waiting another time step, and repeating the cycle. Fig. S7B graphically shows the result of composing the update functions F_4, F_2 and F_0, associated with the proposed scheme, along with their implementation in C#. Note how, indeed, for this update scheme, $\{001\}$ appears as a fixed-point attractor, but when observing the attractor landscape of the synchronous scheme (Fig. S7C), this attractor disappears. With the above we demonstrate that it is possible to design "synchrony-sensitive" attractors by proposing a specific update order.

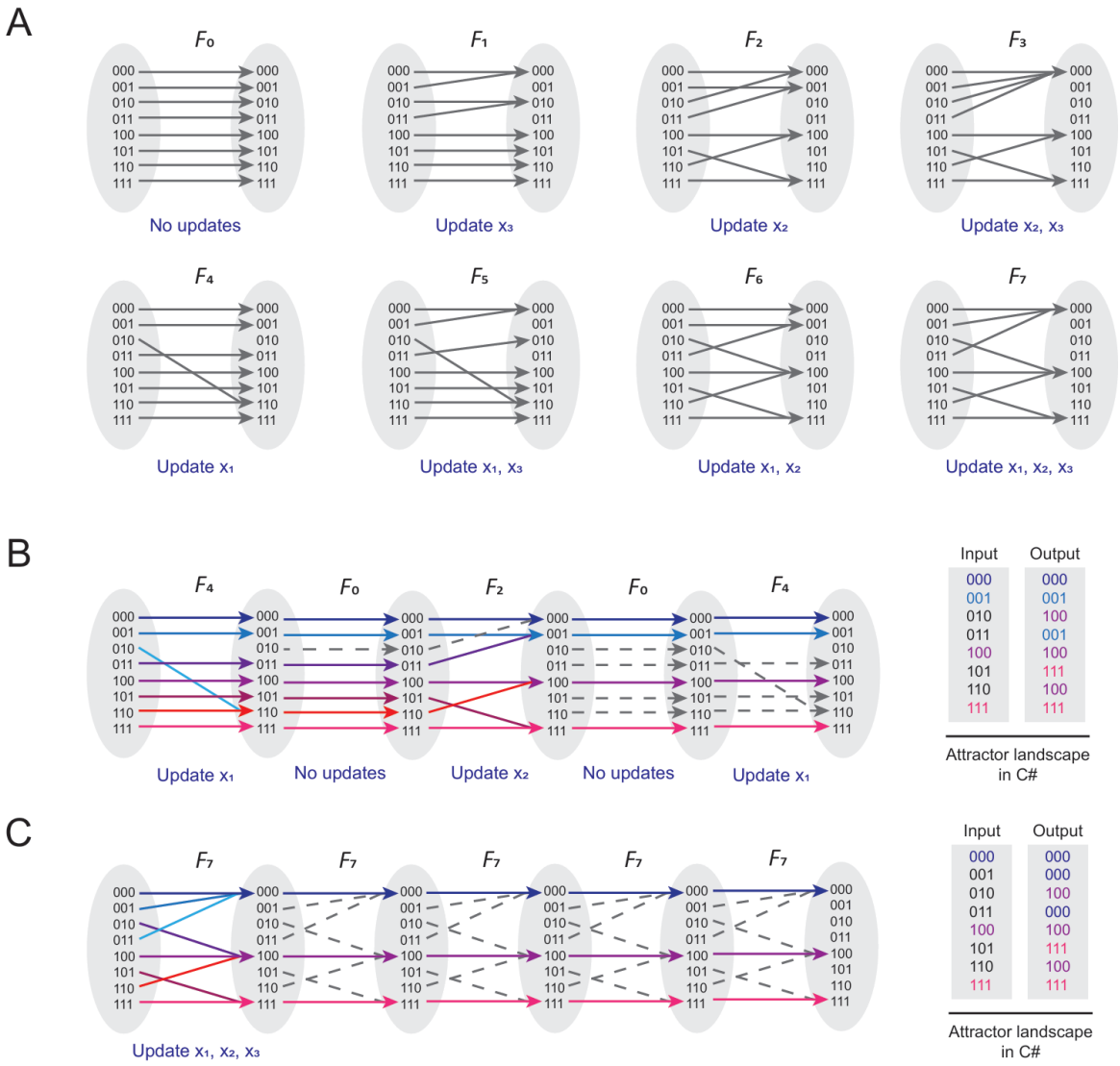

Figure S7. Rational design of a "synchrony-sensitive" attractor. This figure illustrates the procedure for rationally designing a synchronous attractor. (A) First, the system's update functions must be characterized, and those containing the attractor of interest must be identified. In this case, the attractor $\{001\}$ is created, which exists in functions F_0, F_2, F_4, and F_6. (B) Next, an update scheme is proposed that uses the functions containing the attractor of interest. In this case, the proposed composition is F_2, F_0, and F_4, which is equivalent to first updating node x_1, pausing for one time step, and then updating node x_2. It is worth noting that the C# implementation of this asynchronous update scheme generates $\{001\}$ as a fixed-point attractor. (C) Finally, it can be verified that the attractor $\{001\}$ does not exist in the synchronous update scheme, so it is considered a synchrony-sensitive attractor.

Boolean networks are stable in the Lyapunov sense

Similar to other types of dynamical systems, the concept of stability is fundamental to understand the behavior of Boolean networks. In this regard, since the early 2000s, Kauffman *et al*. (24) reported that channeling functions conferred stability to genetic networks. Despite

the interest of this study, their approach excluded other logic functions that are part of other natural genetic networks. We decided to revisit this open question and investigate whether all Boolean networks are stable. By stability, we mean that the trajectories of the Boolean network remain bounded in time and do not go far away, i.s., stability in Lyapunov sense (25). At first glance, visualizing Lyapunov stability in Boolean networks can be difficult, but if we transform the networks into functions contained in the integers, the concept of stability becomes clear. Converting Boolean networks to integers is something that, along with other researchers, we have shown to be feasible and convenient for changing our perspective on Boolean networks (26). Now, we present the formal proof that Boolean networks are stable in the Lyapunov sense.

Theorem 2 (Stability of Boolean networks): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes. If F is redefined in integers as F^*, then all trajectories of F^* are stable in the Lyapunov sense.

Proof: Let $F^*: \mathbb{Z} \to \mathbb{Z}$, we know that F^* has an identical domain and codomain, essentially defined by $\Omega^* = \{0, 1, \dots, 2^n - 1\}$. Let's suppose that there exists an unstable trajectory $r \in F^*$. This implies that $r \to \infty$ as $t \to \infty$. Consequently, r will reach a value greater than its state space maximum element, $2^n - 1$. But we have fallen into a contradiction, since r is the update image of F^*, and this image cannot be projected outside the codomain whose limits are 0 and $2^n - 1$. Therefore, every $r \in F^*$ is bounded to the codomain of F^* at each time step, which implies that it is stable in the Lyapunov sense ■

Remark 3: Note that stability in the Lyapunov sense for Boolean networks apply to all update schemes, even stochastic schemes.

Example 7: To illustrate these concepts, consider the following Boolean network:

$$\begin{cases} x_1^+ = x_2 \\ x_2^+ = x_1 \end{cases} \tag{E. 10}$$

This network has the following associated functions:

$$F_3(x_1^+, x_2^+) = (x_2, x_1)$$

$$F_2(x_1^+, x_2) = (x_2, x_2)$$

$$F_1(x_1, x_2^+) = (x_1, x_1)$$

$$F_0(x_1, x_2) = (x_1, x_2)$$

It worthy to note that, when converting the configuration space of the network F to integers (Fig. S8A), the trajectories of the synchronous (Fig. S8B), asynchronous (Fig. S8C), and stochastic (Fig. S8D) update schemes always remain bounded between 0 and 3, never leaving this region in the set of integers, even over time. Therefore, Boolean networks are stable in the Lyapunov sense, regardless of the update scheme used to study their dynamics.

Boolean networks are not chaotic *per se*

The intrinsic stability of Boolean networks, along with determinism, raises important questions that have been circulating since the early 2000s. For example, it is unknown whether Boolean networks will be chaotic or not. In this regard, some argue that Boolean networks, if they behave like randomly connected cellular automata, may exhibit chaos (27). However, this claim is not entirely applicable to classical Boolean networks, as this approach does not focus on the intrinsic characteristics of such networks (27). Consequently, we investigate whether chaos is actually possible in Boolean networks.

Theorem 3 (Unchaotic dynamics of Boolean networks): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes. If F has a time-invariant structure, then there are no chaotic trajectories in the dynamics of F.

Proof: Suppose the network F has a chaotic path r. It means that the composition of update functions derived from F generates a chaotic path, that is:

$$r(x) = (F_1 \circ F_2 \circ F_3 \circ \ldots \circ F_k)(x)$$

Note that r must be aperiodic and stable. However, since F does not change its structure, this implies that all update functions have a known structure and, consequently, any deterministic update scheme generated from these functions necessarily has a period. Therefore, we have fallen into a contradiction, since the only way to have an aperiodic update chain is through a stochastic update scheme, which is not deterministic. Consequently, chaotic paths do not exist in Boolean networks ■

With this demonstration, it is clear that it is not possible to have chaos in the classical sense of the word in Boolean networks, because the networks that are updated in deterministic schemes are always periodic, a condition contrary to the obligatory aperiodicity of chaos (25) (Fig. S8D).

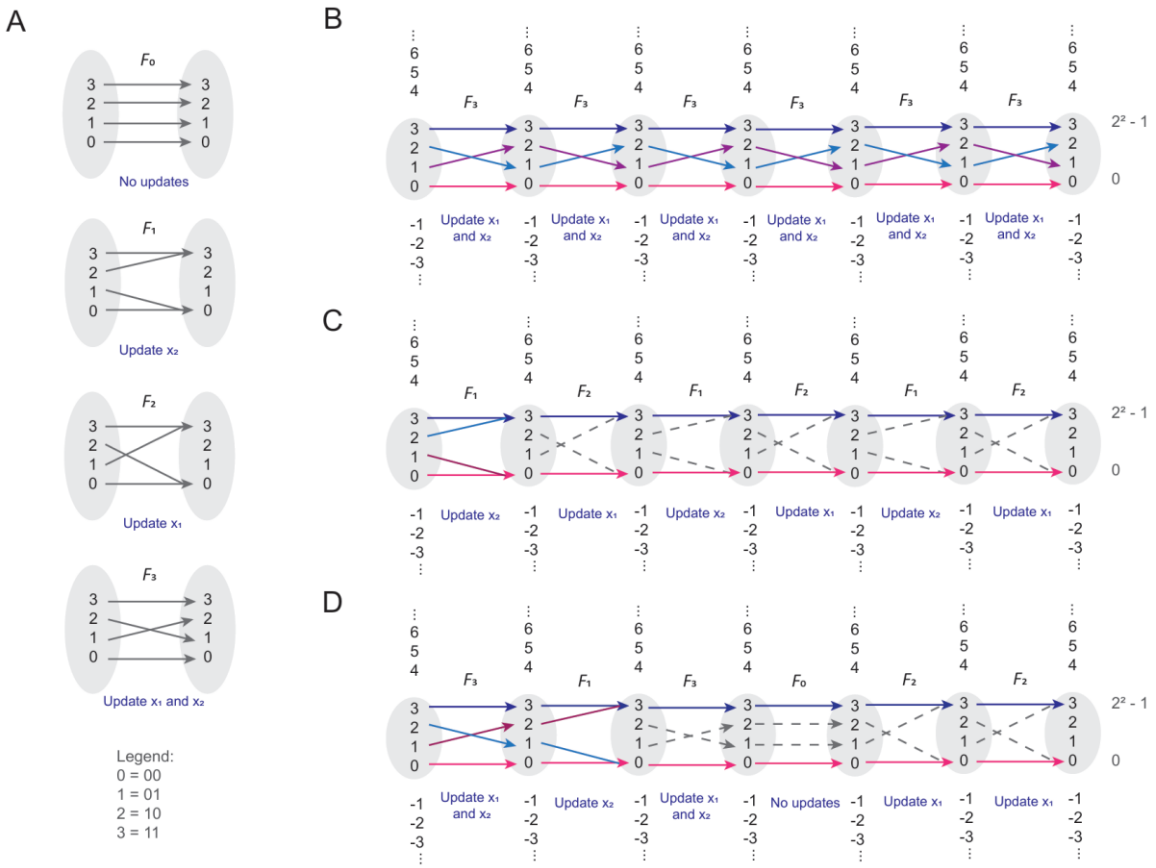

Figure S8. Graphical demonstration of the stability of Boolean networks. This figure graphically demonstrates that Boolean networks are stable in the Lyapunov sense. If the update functions are converted to integers, as shown in panel (A), it is verified that none of the paths leave the bounded space between 0 and $2^n - 1$, regardless of the update scheme used. This can be verified for the synchronous scheme, as shown in panel (B), as well as for the asynchronous scheme (C) and the stochastic scheme. Note that, in this case, $2^n - 1 = 3$.

Temporal control of Boolean networks and design of synchrony-sensitive attractors

The main implication of the fact that "synchrony-sensitive" attractors are local fixed-points of update schemes is that they can be created based on the node update order. Thus, it opens the possibility of controlling the emergence of these attractors. Consequently, it is possible to design update schemes so that the system reaches specific desired attractors. The main theoretical aspects for implementing this technique for controlling the dynamics of Boolean networks are described below.

Lemma 3: Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, and let $F_0, F_1, F_2, F_3, \dots, F_{2^n-1}$ the update functions derived from F. If F is defined, then F_0 has the maximum set of attractors and F_{2^n-1} has the minimum set of attractors of F.

Proof: By Definition 2, F_0 is the conservation function which means that there are no updates. Observe that, in such a case, all initial configurations remain constant. This implies that all 2^n configurations of the state space remain constant, which means that all configurations are fixed-point attractors for this function. On the other hand, by definition F_{2^n-1} corresponds to synchronous update function, which has a set of fixed-point attractors A. By Theorem 1, we know that fixed-points of the synchronous update scheme are global attractors, which implies that there is no update scheme that has less attractors than the synchronous one. Therefore, F_0 has the maximum set of attractors, which is the entire state space, and F_{2^n-1} has the minimum set of attractors ■

Corollary 2: However, $F_1, F_2, \dots F_{2^n-2}$ might have more attractors than the synchronous update function F_{2^n-1} but fewer than the conservation function F_0.

Theorem 4 (Emergence of synchrony-sensitive attractors): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, and let $F_0, F_1, F_2, F_3, \dots, F_{2^n-1}$ the update functions obtained from F. If there are a set of update functions that share fixed-point attractors in addition to the global attractors given by the synchronous update function F_{2^n-1}, then the resulting composition of functions will have global attractors and synchrony-sensitive attractors.

Proof: Suppose F_{2^n-1} has the set of fixed-point attractors $A = \{a_1\}$. By Theorem 1, we already know that fixed-points of the synchronous update scheme appears in every composition of the update functions of F. Now, suppose that exists F_1, F_2, F_3 such that they have the sets of local fixed-point attractors $A_1 = \{a_1, a_2, a_3\}$, $A_2 = \{a_1, a_2, a_4\}$, $A_3 = \{a_1, a_4, a_5\}$ respectively. Applying the same reasoning of Theorem 1, observe that composing $F_1 \circ F_2$ or $F_2 \circ F_1$ generates a set of attractors given by $\{a_1, a_2\}$. Similarly, composing either $F_2 \circ F_3$ or $F_3 \circ F_2$ generates the set of attractors $\{a_1, a_4\}$, but composing $F_1 \circ F_3 \circ F_{2^n-1}$, $f_3 \circ F_1 \circ F_{2^n-1}$, $F_2 \circ F_3 \circ F_{2^n-1}$ or $F_3 \circ F_2 \circ F_{2^n-1}$ always produces the same set of attractors $\{a_1\}$, which is the obtained from the synchronous update scheme. This implies that a_2 and a_4 are synchrony-sensitive attractors, because they disappear after synchrony update, while a_1 is a global attractor given by the synchronous update scheme ■

Theorem 5 (Stability by asynchrony): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, and let $F_0, F_1, F_2, F_3, \ldots, F_{2^n-1}$ the update functions derived from F and let $\beta \in \mathbb{B}^n$ that is not a fixed-point of the synchronous update scheme. If $\beta \in \mathbb{B}^n$ is stabilizable by asynchrony, then exists at least one update function F_k such that $F_k(\beta) = \beta$ and $F_k \neq F_0$.

Proof: Suppose that β is stabilizable by asynchrony and $F_k = F_0$. We must remember that the conservation function F_0 appears when the network F has no updates. Thus, the only way to make β stable is when the network is not updated, which implies that there is no update function able to maintain β stable. But here is a contradiction, since we supposed β is stabilizable by asynchrony, it implies that in certain time, a set of nodes must be updated to maintain β stable ■

Definition 4 (Reachability by asynchrony): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, and let $F_0, F_1, F_2, F_3, \ldots, F_{2^n-1}$ the update functions derived from F. A configuration $x_0 \in \mathbb{B}^n$ is reachable by asynchrony if exists at least one $F_k(x_0) = x_0$ such that $F_k = F_0$.

Definition 5 (Test of reachability by asynchrony): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, and let $F_0, F_1, F_2, F_3, \ldots, F_{2^n-1}$ the update functions derived from F. To test the reachability by asynchrony property of a configuration $x_0 \in \mathbb{B}^n$ which is not an attractor of the synchronous update scheme, we shall look for a set of update functions such that $F_j(x_0) = x_0$ obtained by direct substitution in F_j.

For example, continuing with the network E. 8, if we want to investigate whether (01) or (11) are reachable by asynchrony, we need to substitute these values at logical rules as follows:

$$x_1^+ = x_2 + \bar{x}_2 = 0$$
$$x_2^+ = \bar{x}_1 x_1 = 1$$

The only way to do this possible, occurs when

$$x_1^+ = x_1$$
$$x_2^+ = x_2$$

Which is by definition the conservation function. Thus, (01) is not stabilizable by asynchrony. On the other hand, for (11) we observe that

$$x_1^+ = x_2 + \bar{x}_2 = 1$$
$$x_2^+ = \bar{x}_1 x_1 = 1$$

And the only way to do this, occurs when

$$x_1^+ = x_2 + \bar{x}_2$$
$$x_2^+ = x_2$$

Which means that (11) can be stabilized by asynchrony, as it can be observed in Fig. S6A.

Definition 6 (Temporal control of dynamics): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, and let $F_0, F_1, F_2, F_3, \ldots, F_{2^n-1}$ the update functions derived from F. The temporal control of F, occurs when it is proposed a composition of update functions such that, the dynamics of F reaches a specific set of desired fixed-points, which includes synchrony-sensitive attractors in a defined time-lapse. To implement this control, it is mandatory execute the node update that is represented by the update functions in the order they were composed. It is noteworthy to say that, composing a single function is equivalent to advancing one time-step only.

Definition 7 (Temporal controllability): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, and let $x_0, x_f \in \mathbb{B}^n$. The network F is fully controllable if for each initial condition $x_0 \in \mathbb{B}^n$ there is a composition of update functions H of the network F, such that $H(x_0) = x_f$. If this property is partially satisfied by a network, we shall say that it is partially controllable. Otherwise, the network is temporarily uncontrollable.

To illustrate these principles, let's see the following example:

Example 8: Consider the Boolean network

$$\begin{cases} x_1^+ = x_1 x_2 \\ x_2^+ = x_1 x_2 \\ x_3^+ = x_1 x_2 + x_3 \end{cases} \quad \text{(E. 11)}$$

From this network, we can obtain its update functions as:

$$F_0(x_1, x_2, x_3) = (x_1, x_2, x_3)$$

$$F_1(x_1, x_2, x_3^+) = (x_1, x_2, x_1 x_2 + x_3)$$

$$F_2(x_1, x_2^+, x_3) = (x_1, x_1 x_2, x_3)$$

$$F_3(x_1, x_2^+, x_3^+) = (x_1, x_1 x_2, x_1 x_2 + x_3)$$

$$F_4(x_1^+, x_2, x_3) = (x_1 x_2, x_2, x_3)$$

$$F_5(x_1^+, x_2, x_3^+) = (x_1 x_2, x_2, x_1 x_2 + x_3)$$

$$F_6(x_1^+, x_2^+, x_3) = (x_1 x_2, x_1 x_2, x_3)$$

$$F_7(x_1^+, x_2^+, x_3^+) = (x_1 x_2, x_1 x_2, x_1 x_2 + x_3)$$

After solving these update functions (Fig. S9A), note that for all update functions $\{(000), (001), (111)\}$ are global attractors, and for F_1, F_4, F_5 there are $\{(010), (011)\}$ as local attractors. Similarly, local fixed-points $\{(100), (101)\}$ and $\{(110)\}$ emerge for compositions of $F_1, F_2 F_3$ and F_2, F_4, F_6 respectively. Therefore, if we propose the asynchronous update scheme $x_3 \to x_1 \to x_1, x_3$ or any combination that does not update x_2, we will be able to stabilize $\{(010), (011)\}$ as local attractors (Fig. S9B). This is equivalent

to produce any compositions of F_1, F_4, F_5. In this way, any update order that do not consider x_1 is equivalent to make any composition of F_1, F_2, F_3which it will stabilize $\{(100), (101)\}$ as local attractors (Fig. S9C).

Observe that, the creation of synchrony-sensitive attractors can also be transitory. For instance, if we propose the update order: $x_1 \rightarrow x_3 \rightarrow no\ update \rightarrow x_1 \rightarrow x_3 \rightarrow x_2$, which implies that $\{(010), (011)\}$ will be stable only for five time-steps, and they will disappear for the sixth time-step (Fig. S9D). This principle can be applied to extend the duration of any state of the network E. 11. However, once local attractors disappear, there is not possible to make them stable again (Fig. S9D). Another important limitation of this approach is that, there are states that cannot be stabilized by asynchrony. For example, observe the network E. 8 and Fig. S6. In this network the configuration $(0,1)$ does not appear as local attractor at any update function (Fig. S6A), except by F_0, and by Theorem 5, this implies that $(0,1)$ cannot be stabilized by this methodology. Consequently, by Definition 7, this network is partially controllable, because there are compositions of update function that allow to reach all states, except (0,1).

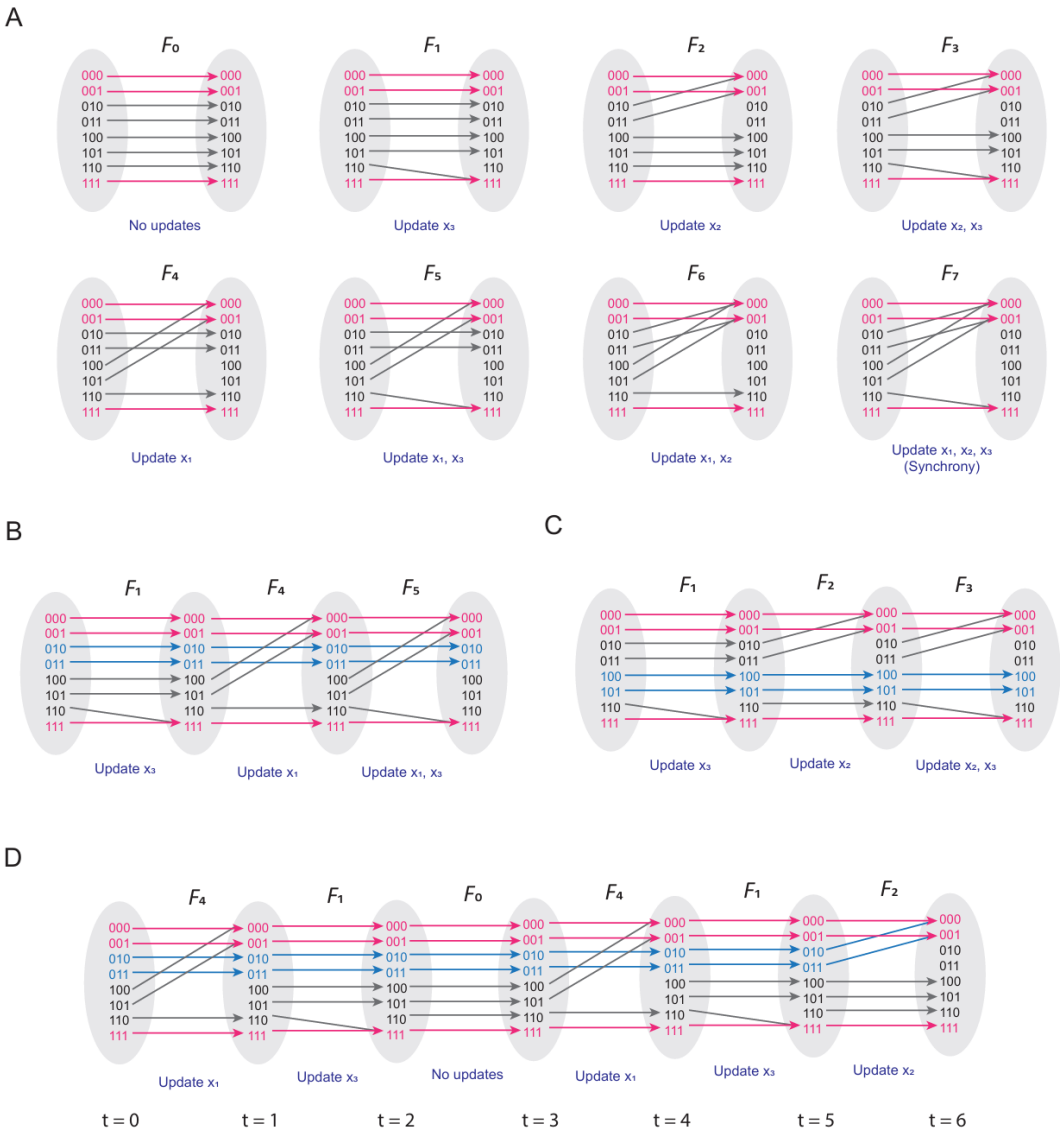

Figure S9. Example of temporal control of a network. This figure shows how to design an asynchronous update scheme with specific dynamical features. (A) The network E. 11 produce eight different update functions, which has global attractors in pink and local attractors in grey straight lines. (B) If we used this functions to propose a specific asynchronous update scheme, such as $x_3 \rightarrow x_1 \rightarrow x_1, x_3$, it will produce "synchrony-sensitive" or local attractors, such $\{(010), (011)\}$ in blue. (C) The same principle applies to other update schemes such as $x_3 \rightarrow x_2 \rightarrow x_2, x_3$, which stabilizes $\{(100), (101)\}$ as local attractors. (E) Finally, the stabilization of any local attractor can be designed. For instance, in the update scheme $x_1 \rightarrow x_3 \rightarrow no\ update \rightarrow x_1 \rightarrow x_3 \rightarrow x_2$, the local attractors $\{(010), (011)\}$ remain stable for 5 time-steps, but for the 6^{th} time-step, they disappear and are absorbed by global fixed-points (000) and (001). In this way, the stabilization time of a local attractor can be programmed.

Structural control of Boolean networks and rational design of dynamics

The previous control method shows us that it is possible to design an asynchronous update scheme to achieve a desired behavior within a specific timeframe. Furthermore, this approach reveals that there are states that, by their very nature, cannot be stabilized in any way. However, the question remains whether there is a way to overcome these limitations, which depend entirely on the network structure. To find a control method that allows us to modify the structure of Boolean networks, let's first examine Kauffman's original model for a network of n nodes:

$$\begin{cases} x_1^+ = f_1(x_1, x_2, \dots, x_n) \\ x_2^+ = f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ x_n^+ = f_n(x_1, x_2, \dots, x_n) \end{cases}$$

In this model, each updated state x_i^+ is calculated by its logic rule $f_i(x_1, x_2, \dots, x_n)$ which is a function of all nodes of the network and $f_i: \mathbb{B}^n \to \mathbb{B}$. As we have seen in previous sections, changing the notation of the classic Kauffman model can give us a different perspective on the properties of Boolean networks. So, this time, we will look at Boolean networks and the Kauffman model from a vector perspective. That is:

$$\begin{bmatrix} x_1^+ \\ x_2^+ \\ \vdots \\ x_n^+ \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) \end{bmatrix} = \begin{bmatrix} a_{11}m_0 + a_{12}m_1 + \cdots a_{1(2^n)}m_{(2^n-1)} \\ a_{21}m_0 + a_{22}m_1 + \cdots a_{2(2^n)}m_{(2^n-1)} \\ \vdots \\ a_{n1}m_0 + a_{n2}m_1 + \cdots a_{n(2^n)}m_{(2^n-1)} \end{bmatrix}$$

Note that each logical rule can be expressed as a polynomial, expressed in disjunctive normal form. Each polynomial corresponds to the sum of minterms, that is, combinations of the state space for which the logical rule is true. Factoring we have:

$$\begin{bmatrix} a_{11}m_0 + a_{12}m_1 + \cdots a_{1(2^n)}m_{(2^n-1)} \\ a_{21}m_0 + a_{22}m_1 + \cdots a_{2(2^n)}m_{(2^n-1)} \\ \vdots \\ a_{n1}m_0 + a_{n2}m_1 + \cdots a_{n(2^n)}m_{(2^n-1)} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1(2^n)} \\ a_{21} & a_{22} & \cdots & a_{2(2^n)} \\ & & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{n(2^n)} \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{(2^n-1)} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1(2^n)} \\ a_{21} & a_{22} & \cdots & a_{2(2^n)} \\ & & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{n(2^n)} \end{bmatrix} \begin{bmatrix} \bar{x}_1\bar{x}_2 \dots \bar{x}_{(n-1)}\bar{x}_n \\ \bar{x}_1\bar{x}_2 \dots \bar{x}_{(n-1)}x_n \\ \vdots \\ x_1x_2 \dots x_{(n-1)}x_n \end{bmatrix}$$

Now, observe that each minterm can be obtained by the semi-tensorial product of vectors with the form of $\left[\bar{x}_j, x_j\right]^T$, in other words:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1(2^n)} \\ a_{21} & a_{22} & \cdots & a_{2(2^n)} \\ & \vdots & & \\ a_{n1} & a_{n2} & \cdots & a_{n(2^n)} \end{bmatrix} \begin{bmatrix} \bar{x}_1\bar{x}_2 \ldots \bar{x}_{(n-1)}\bar{x}_n \\ \bar{x}_1\bar{x}_2 \ldots \bar{x}_{(n-1)}x_n \\ \vdots \\ x_1x_2 \ldots x_{(n-1)}x_n \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1(2^n)} \\ a_{21} & a_{22} & \cdots & a_{2(2^n)} \\ & & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{n(2^n)} \end{bmatrix} \left[\begin{bmatrix} \bar{x}_1 \\ x_1 \end{bmatrix} \ltimes \begin{bmatrix} \bar{x}_2 \\ x_2 \end{bmatrix} \ltimes \ldots \ltimes \begin{bmatrix} \bar{x}_n \\ x_n \end{bmatrix} \right] = M \prod_{k=1}^{n} \begin{bmatrix} \bar{x}_k \\ x_k \end{bmatrix} = M\vec{\omega}$$

Therefore, the Kauffman's model can be expressed in vectorial form as:

$$\overrightarrow{x^+} = M\vec{\omega}$$

Where M is a matrix of $2^n \times n$, and $\vec{\omega}$ is the entire state-space expressed as a vector. Note that M is a matrix of constant coefficients $a_{ij} \in \mathbb{B}$ and its position is defined by the logic rule *per se*.

Theorem 6 (Vectorial equivalence to update functions): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of *n* nodes. If F is written as $\overrightarrow{x^+} = M\vec{\omega}$, then for all configuration $r_k \in \mathbb{B}^n$ exists a column vector $\vec{v} \in M$, such that $\vec{v} = F(r_k)$.

Proof: We know that a Boolean network F of *n* nodes is constructed by *n* logic rules, i.e.,

$$F(x_1, x_2, \ldots, x_n) = \begin{cases} x_1^+ = f_1(x_1, x_2, \ldots, x_n) \\ x_2^+ = f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ x_n^+ = f_n(x_1, x_2, \ldots, x_n) \end{cases}$$

We also know that each logic rule $f_i: \mathbb{B}^n \to \mathbb{B}$, can only give $a_i \in \{0,1\}$ as output, depending on its definition. Now, if we evaluate $F(r_k)$, then

$$F(r_k) = \begin{cases} x_1^+ = f_1(r_k) = a_{1k} \\ x_2^+ = f_2(r_k) = a_{2k} \\ \vdots \\ x_n^+ = f_n(r_k) = a_{nk} \end{cases}$$

Observe that $\vec{v}^T = [a_{1k}, a_{2k}, \ldots, a_{nk}]^T \in M$. Therefore, for all $r_k \in \mathbb{B}^n$, there is $\vec{v} \in M$, such that $\vec{v} = F(r_k)$ ■

Corollary 3: Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of *n* nodes and $\alpha \in \mathbb{B}^n$. If F is written as $\overrightarrow{x^+} = M\vec{\omega}$ and $F(\alpha) = \alpha$, then $\alpha \in M$.

Definition 8 (Structural control of Boolean dynamics): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes. The structural control of the dynamics of F occurs when this network is written as $\overrightarrow{x^+} = M\vec{\omega}$ and the matrix $M_{n\times 2^n}$ is modified by a control law $u_{n\times 2^n}$ as follows: $M \oplus u$. This allows to create any dynamical behavior for the network in a specific time-lapse. Where "$\oplus$" is the XOR operator. To implement this control, first design an $u_{n\times 2^n}$, then execute $M \oplus u$, and after that, simplify the resulting matrix.

Remark 4: Observe that, if a target dynamic T is set such that $M \oplus u = T$, the control law u can be designed by setting $u = M \oplus T$.

To illustrate these concepts, let's see the following example.

Example 9: Consider the Boolean network

$$\begin{cases} x_1^+ = x_1 \\ x_2^+ = x_1 x_2 \\ x_3^+ = x_1 + \bar{x}_1 x_2 x_3 \end{cases} \quad \text{(E. 12)}$$

Transforming this network in its disjunctive normal form, we obtain:

$$\begin{cases} x_1^+ = x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2 x_3 + x_1 x_2\bar{x}_3 + x_1 x_2 x_3 \\ x_2^+ = x_1 x_2\bar{x}_3 + x_1 x_2 x_3 \\ x_3^+ = \bar{x}_1 x_2 x_3 + x_1\bar{x}_1\bar{x}_1 + x_1\bar{x}_2 x_3 + x_1 x_2\bar{x}_3 + x_1 x_2 x_3 \end{cases}$$

In vectorial form, we obtain:

$$\begin{bmatrix} x_1^+ \\ x_2^+ \\ x_3^+ \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \bar{x}_1\bar{x}_2\bar{x}_3 \\ \bar{x}_1\bar{x}_2 x_3 \\ \bar{x}_1 x_2\bar{x}_3 \\ \bar{x}_1 x_2 x_3 \\ x_1\bar{x}_2\bar{x}_3 \\ x_1\bar{x}_2 x_3 \\ x_1 x_2\bar{x}_3 \\ x_1 x_2 x_3 \end{bmatrix}$$

Thus, the matrix M is given by:

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The most important implication of Theorem 6 is that each column vector of the matrix corresponds to a specific minterm, which means that from this matrix we can reconstruct the dynamics of the Boolean network as follows:

$$minterm\ as\ input\ (000) \rightarrow [0,0,0]^T\ vector\ as\ output$$
$$(001) \rightarrow [0,0,0]^T$$
$$(010) \rightarrow [0,0,0]^T$$
$$(011) \rightarrow [0,0,1]^T$$
$$(100) \rightarrow [1,0,1]^T$$
$$(101) \rightarrow [1,0,1]^T$$
$$(110) \rightarrow [1,1,1]^T$$
$$(111) \rightarrow [1,1,1]^T$$

Observe this representation is equivalent to the synchronous update function, and represents the entire dynamics of network E.12. As it can be seen, this network has three global fixed-points given by $\{(000), (101), (111)\}$ (Fig. S10A). Now suppose that, as a control objective, we want to stabilize configuration (001) as a global attractor of the dynamics. To achieve this objective, we need to add the following control law to the system matrix M:

$$u_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In other words:

$$M + u_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

After this modification, we can reconstruct the network E. 12 as follows:

$$\begin{cases} x_1^+ = x_1 \\ x_2^+ = x_1 x_2 \\ x_3^+ = x_1 + \bar{x}_1 x_2 x_3 + \bar{x}_1 \bar{x}_2 x_3 \end{cases}$$

Note that the term $\bar{x}_1 \bar{x}_2 x_3$ at x_3^+ is the perturbation that we need to implement in order to make (001) a fixed-point attractor (Fig. S10B). Now, suppose that the control objective is to destroy the fixed-point (111) from the original dynamics and direct its trajectories to the attractor (101). To implement this modification, we need to use the following control law:

$$u_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Then, we need to modify M as follows:

$$M \oplus u_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

After this modification, the modified network E. 12 is:

$$\begin{cases} x_1^+ = x_1 \\ x_2^+ = x_1 x_2 \bar{x}_3 \\ x_3^+ = x_1 + \bar{x}_1 x_2 x_3 \end{cases}$$

In this new network, the term $x_1 x_2 \bar{x}_3$ guarantees that (111) is not a fixed-point and sends its trajectories to (101) (Fig. S10C). This control technique is capable of finely modifying the dynamics of a network; for example, it is possible to specify the transients that will be reached within a defined time-lapse (Fig. S10D, E). Similarly, it is possible to create cyclic attractors (Fig. S10F), modify cyclic attractors (Fig. S10G), destroy cyclic attractors (Fig. S10H), and even reverse the trajectories followed by the dynamics (Fig. S10D). In other words, this control method has no limits for designing Boolean networks as mathematical objects.

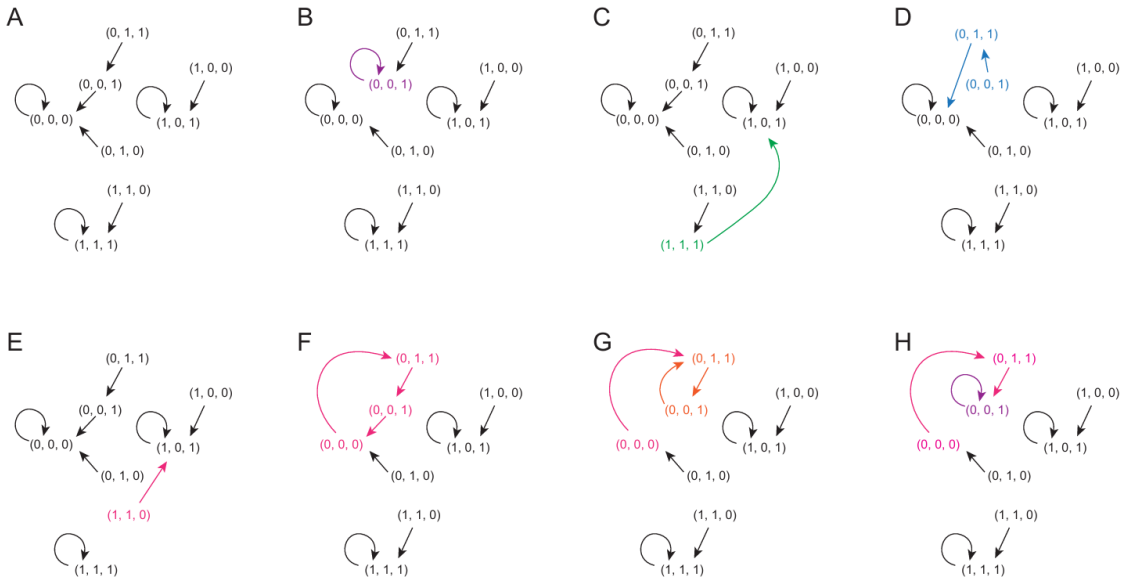

Figure S10. Example of structural control of a network. This figure shows some of the possibilities to design the dynamics of the network E. 12. (A) Original dynamics of network E. 12. (B) Creation of a fixed-point attractor. (C) Destruction of a fixed-point attractor. (D) Modification of transitory states of a trajectory. (E) Modification of a trajectory. (F) Creation of a cyclic attractor. (G) Redefinition of a cyclic attractor. (H) Destruction of a cyclic attractor.

However, this does not mean that the method is infallible for controlling biological systems. It is important to mention that some biological systems can be permanently modified through gene editing, while others may be transient due to the presence of actuators such as multifunctional proteins or antibodies, or through drugs. In short, this method of structural control is limited by the feasibility of interventions in real biological systems. To illustrate this principle, let's look at the following example.

Example 10: Consider the Boolean network

$$\begin{cases} ATR^{+} = ATR \\ p53^{+} = ATR \wedge \neg Mdm2 \\ Mdm2^{+} = p53 \end{cases} \quad \text{(E. 13)}$$

This network represents the activation circuit of the tumor suppressor p53. Under conditions of genotoxic damage, kinases such as ATR are activated and phosphorylate several proteins, including the transcription factor p53. Once p53 is phosphorylated, it activates its function and begins transcribing several genes to arrest the cell cycle, repair genotoxic damage, and prevent the death of the damaged cell. At the same time, it also induces the expression of its repressor Mdm2, which functions to promote the degradation of p53 in order to maintain homeostatic control of its levels in the cell (1) (Fig. S11A). Let's imagine that, as a control objective, we want ATR to be permanently activate without p53 being activated. To do this, we will first change the notation of the biological network as follows:

$$\begin{cases} x_1^+ = x_1 \\ x_2^+ = x_1\bar{x}_3 \\ x_3^+ = x_2 \end{cases}$$

Where x_1 is ATR, x_2 is p53 and x_3 is Mdm2. Next, we will obtain the disjunctive normal form of the network, that is:

$$\begin{cases} x_1^+ = x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3 \\ x_2^+ = x_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3 \\ x_3^+ = \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3 \end{cases}$$

From this, we can obtain the system matrix given by:

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

From this matrix, we can understand the entire dynamics of the synchronously updated network, which is given by:

$$\begin{gathered} (000) \to [0,0,0]^T \\ (001) \to [0,0,0]^T \\ (010) \to [0,0,1]^T \\ (011) \to [0,0,1]^T \\ (100) \to [1,1,0]^T \\ (101) \to [1,0,0]^T \\ (110) \to [1,1,1]^T \\ (111) \to [1,0,1]^T \end{gathered}$$

This means that there is a fixed-point attractor at (000), and a cycle at $(100) \to (110) \to (111) \to (101) \to (100)$ (Fig. S11B). As part of the control objective, we specified that ATR should always remain active without p53. Therefore, it is necessary to destroy the attractor (000) to ensure that ATR remains active, and we must inhibit the activation cycle of p53. Preferably, we should guide the dynamics toward state (101). To do this, we will use the following control law:

$$u = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Now, let's study the new dynamics that result from the addition of the control law:

$$M \oplus u = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \oplus \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The resulting dynamics of this new network can be observed in Fig S11C. Therefore, the new network resulting is given by:

$$\begin{cases} x_1^+ = x_1 + \bar{x}_1\bar{x}_2\bar{x}_3 \\ x_2^+ = x_1\bar{x}_3 \\ x_3^+ = x_2 + x_1\bar{x}_2x_3 \end{cases}$$

Observe that the term $\bar{x}_1\bar{x}_2\bar{x}_3$ implies that ATR must be activated in absence of p53, ATR and Mdm2. Similarly, the term $x_1\bar{x}_2x_3$ indicates that Mdm2 must be activated in absence of p53 but in presence of Mdm2 and ATR. To implement these exact conditions in real cells, it is necessary to genetically modify the network so that Mdm2 is overexpressed and ATR is self-activated. This does not seem practically feasible, because it requires to induce two potentially lethal mutations at the same time. However, using drugs or viral multifunctional proteins, it is possible to achieve simultaneous self-activation of ATR and Mdm2. In this sense, this mechanism occurs precisely in nature. Specifically, in the case of SARS-CoV-2 infection, the S protein is known to activate ATR (2), while the viral protein PLP activates the expression of Mdm2 (3). In other words, the network can be controlled as follows:

$$\begin{cases} x_1^+ = x_1 + S \\ x_2^+ = x_1\bar{x}_3 \\ x_3^+ = x_2 + PLP \end{cases}$$

Where $S, PLP = 1$ and its dynamics is shown in Fig. S11D. Thus, although the dynamics are not exactly as the design indicated, it is possible to implement this type of control in a real biological context. The most notable about this approach is that it's conceptually similar to the pole design of classical control theory. This is because, given a particular dynamic design, the mathematical conditions emerge that allow for the implementation of the control law. Furthermore, this new approach enables fine-tuning of the dynamics and their behavior over time. However, it will always be limited by the feasibility of implementation in real biological systems. Therefore, it is mandatory to add the following concept

Definition 9 (Structural reachability): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, expressed as $\overrightarrow{x^+} = M\vec{\omega}$, and let u a control law. A particular state or dynamics can be reached in $\overrightarrow{x^+}$ by doing $M \oplus u$ if there are no structural impediments in the network to implement the control law.

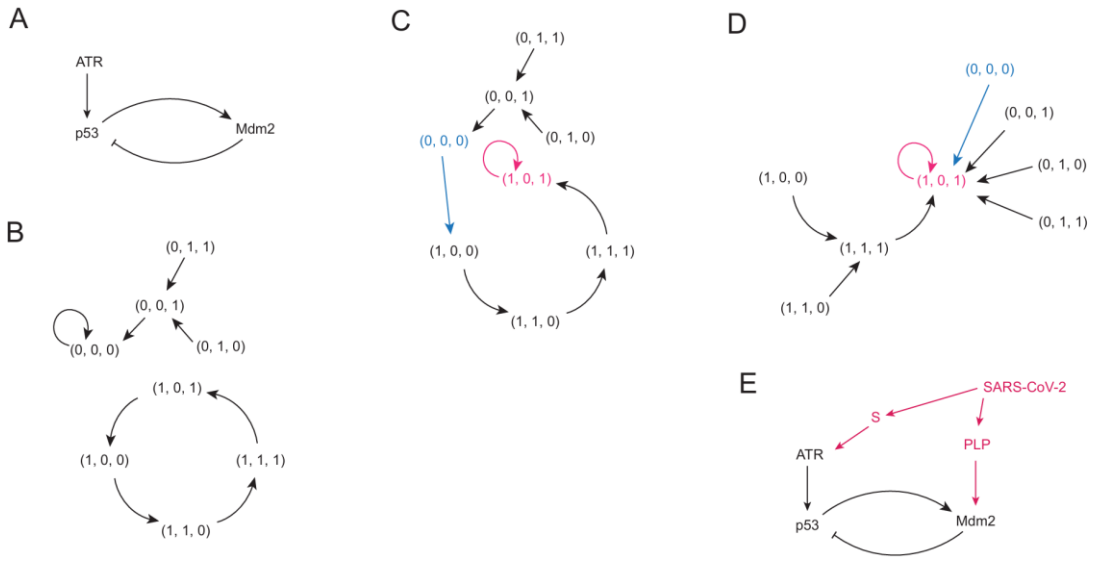

Figure S11. Example of structural control of a biological network. This figure shows how a real biological network can be manipulated. (A) p53-Mdm2 regulatory circuit under basal conditions. (B) Dynamics of the p53-Mdm2 regulatory circuit. Note the cyclic attractor and the single fixed-point attractor. (C) Proposed dynamics, which involve introducing two mutations that are likely to be deleterious under biological conditions. (D) Alternative dynamics where genetic modifications are avoided, but molecules such as drugs or viral multifunctional proteins are used. (E) Actual regulation of SARS-CoV-2 on this circuit. Note that S and PLP effect the dynamics proposed in the previous panel, demonstrating the feasibility of this method for controlling real biological networks.

Definition 10 (Structural controllability): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, expressed as $\overrightarrow{x^+} = M\vec{\omega}$, and let u a control law. The network F is structurally controllable, if for each initial condition $\vec{x}_0 \in \mathbb{B}^n$ there is a desired final condition $\vec{x}_f \in \mathbb{B}^n$ such that $\vec{x}_f = (M \bigoplus u)\vec{x}_0$. If the network F satisfies in parts this criterion, then the network is partially controllable by the structural approach. In case that no initial condition can be directed to a desired state, the network is uncontrollable. Consider this type of control occurs in one time-step.

It is noteworthy to say that restriction of structural controllability depends on the technical nature of the target network. This implies that, there are no analytical form to determine the controllability, except by understanding the technical limitations of the system of interest. For instance, whether a gene regulatory network allows permanent modifications such as knockouts, over-expressions or all kind of modifications, then such a network can be structurally controllable. If this hypothetical network only allows modification at certain nodes or by doing specific manipulations such as knockdowns, then this network is partially structurally controllable. But if the network does not allow modification in any of its nodes, or by any feasible intervention, then it is uncontrollable by the structural approach.

Hybrid control of Boolean networks

As we said earlier, there are some networks that cannot be manipulated by the temporal control method, and sometimes the structure of such networks does not allow a complete structural controllability. It is possible that these networks only permit a transitory manipulation of their structure. For these cases, we developed the following control method:

Definition 11 (Hybrid temporal and structural control): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes that cannot be fully controlled by temporal or structural methods, and let $x_0, x_f \in \mathbb{B}^n$. If F only admits control at certain time-lapses or on a specific set of nodes, then we can use a temporal regime of control to design a trajectory that starts in x_0 and reaches x_f by implementing structural control on the unrestricted nodes as follows:

$$\big(F_i \circ (F_k \oplus u_k) \circ F_j\big)(x_0) = x_f$$

Note that time is given by the length of the chain of compositions of $\big(F_i \circ (F_k \oplus u_k) \circ F_j\big)$, as occurred with our temporal control method. Also, observe that the structural modification is used only on specific time-lapses, which is equivalent to perform asynchronous control updates.

To illustrate how this method works, let's see the following example.

Example 11: Consider the following network, given by the expressions:

$$\begin{cases} x_1^+ = x_1 + \bar{x}_1 \\ x_2^+ = \bar{x}_2 x_2 \end{cases} \quad \text{(E. 14)}$$

As we said earlier, the configuration (0,1) is not reachable by temporal control, since there are no update functions that originate this configuration as an attractor, except the conservation function. Applying the reachability test given by Definition 5, we need to substitute the desired state on the logical rules as follows:

$$x_1^+ = x_1 + \bar{x}_1 = 0$$
$$x_2^+ = \bar{x}_2 x_2 = 1$$

Then, observe that the only way to achieve this, is by making

$$x_1^+ = x_1$$
$$x_2^+ = x_2$$

Which is the conservation function, and thus, there are no combination of updates that originate such a configuration. In other words, (0,1) is not temporarily reachable. Now, suppose the network only allows to modify a single node at time. In this case, we might use structural control on an update function. To implement such a control, we need to know the update functions of the network (Fig. S12A), which are given by:

$$F_3(x_1^+, x_2^+) = (x_1 + \bar{x}_1, \bar{x}_2 x_2)$$

$$F_2(x_1^+, x_2) = (x_1 + \bar{x}_1, x_2)$$

$$F_1(x_1, x_2^+) = (x_1, \bar{x}_2 x_2)$$

$$F_0(x_1, x_2) = (x_1, x_2)$$

Observe that for F_1 and F_2 we might propose control laws u_1, u_2 as follows:

$$M_{F_1} \oplus u_1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$M_{F_2} \oplus u_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \oplus \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Thus, such control laws can be implemented as:

$$F_1(x_1, x_2^+) \oplus u_1 = (x_1, \bar{x}_2 x_2 + u_1)$$

$$F_2(x_1^+, x_2) \oplus u_2 = \left((x_1 + \bar{x}_1)\bar{u}_2, x_2\right)$$

Observe that, if we only use $F_1(x_1, x_2^+) \oplus u_1$ we will have two fixed-points given by $\{(0,1), (11)\}$. Similarly, if we only use $F_2(x_1^+, x_2) \oplus u_2$ we will have other two attractors given by $\{(0,0), (0,1)\}$ (Fig. S12A). Now, by combining $F_1(x_1, x_2^+) \oplus u_1$ and $F_2(x_1^+, x_2) \oplus u_2$ we will have a global attractor in $(0,1)$ and all trajectories converge to them (Fig. S12B). It implies that, with this approach we are able to reconfigure the basins of attraction at will. (Fig. S12B). Therefore, a controller for this network could update node x_2 with $u_1 = 1$ and then x_1 with $u_2 = 1$, to make that all trajectories converge to (0,1) as a global attractor.

Despite the promising of this hybrid approach, it still has limitations. For instance, consider the network E. 8 instead of network E. 14, that is:

$$x_1^+ = x_2 + \bar{x}_2$$
$$x_2^+ = \bar{x}_1 x_1$$

If we use the same conditions of analysis described for E. 14, we will note that, to force the dynamics to (0,1), it requires to manipulate two nodes at the same time, which is prohibited. Thus, network E. 8 under these conditions is not controllable. Therefore:

Definition 12 (Reachability by hybrid approach): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes that cannot be fully controlled by temporal or structural methods, and let $x_0, x_f \in \mathbb{B}^n$. A configuration x_f is reachable from an initial condition x_0 by hybrid control, if exists a composition of controlled update functions such that:

$$\left(F_i \circ (F_k \oplus u_k) \circ F_j\right)(x_0) = x_f$$

And the control interventions do not violate structural restrictions. Otherwise, x_f is not reachable.

Definition 13 (Controllability by hybrid approach): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes that cannot be fully controlled by temporal or structural methods, and let $x_0, x_f \in \mathbb{B}^n$. A network F is controllable by hybrid approach if for each initial condition x_0 there is a composition of controlled update functions that do not exceed structural restrictions such that

$$\left(F_i \circ (F_k \oplus u_k) \circ F_j\right)(x_0) = x_f$$

In this case, the time is given by the length of update functions composed, where each function is equivalent to one time-step. If there are states that cannot be reached by hybrid method, then F is partially controllable. Otherwise, F is uncontrollable.

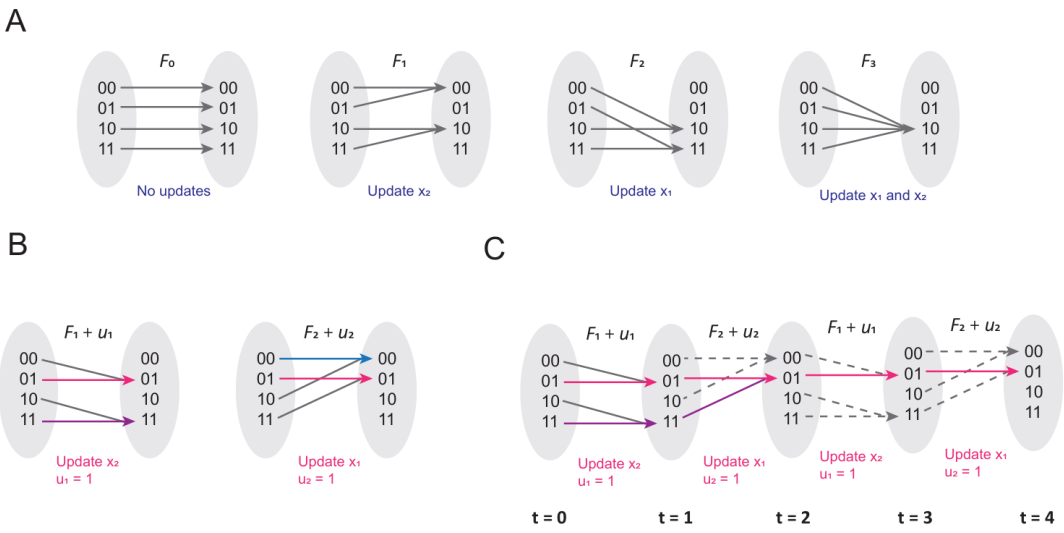

Figure S12. Example of hybrid control of a network. This figure shows how to use hybrid method to control and design a dynamic. (A) update functions of network E. 14., note that (0,1) is not a local attractor for any update function. (B) controlled update functions of network E. 14, observe that by applying the hybrid control method, the desired attractor (0,1) emerges as local attractor. (C) after implementing the asynchronous update scheme with control pulses, observe that all trajectories of the network converge to (0,1), making this state as a global attractor stabilized by asynchrony. Dotted lines represent trajectories that are not visible by the image of the update scheme.

Scalability of control methods

The final point regarding the control methods presented in this work is to determine whether or not they are scalable. In this regard, we previously explored an approach based on partitioning a Boolean network into smaller modules, originally proposed by Madalena Chaves (4) and extended by us in previous work (5). Interestingly, we observed that fixed-point attractors always appeared in the network modules as attractors of those modules, even without all their components. One question that remained unanswered in that preliminary exploration was why fixed-point attractors continued to appear in the modular subnetworks obtained by partitioning a Boolean network. Motivated by this question, we found a logical explanation, which is presented below:

Theorem 7 (Partition of Boolean networks): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of n nodes, and let $\alpha \in \mathbb{B}^n$, such that $F(\alpha) = \alpha$. If the set of logical rules of F given by $f_1, f_2, \dots, f_n$ is partitioned into arbitrary modular subnetworks $S_1, S_2, \dots, S_m$ such that $S_1 \cup S_2 \cup \dots \cup S_m = F$ and $S_1 \cap S_2 \cap \dots \cap S_m = \emptyset$, the components of $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ will be local attractors of each subnetwork, as long as the variables that do not belong to each subnetwork remain fixed.

Proof: Suppose the network $F = (f_1, f_2, \dots, f_n)$ is partitioned into subnetworks $S_1, S_2, \dots, S_m$ that do not share and repeat logical rules, for example: $S_1 = (f_1, f_2)$ and $S_2 = (f_3, f_4)$. Depending on the definition of each logical rule, there will be nodes that will appear in the subnetworks and others do not. If we assume the nodes that do not appear in the logical rules $c_j \in \mathbb{B}$ are constant and the nodes defined in the partitions are denoted by x_j, we can rewrite each subnetwork as $S_k = g_k(x_i; c_j)$, where g_k is a vectorial function of x_i nodes with c_j as parameters. If $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ is a fixed-point for F, this implies that $f_1(\alpha) = \alpha_1, f_2(\alpha) = \alpha_2, \dots, f_n(\alpha) = \alpha_n$. Then, it's easy to see that for each subnetwork $S_k = g_k(x_i; c_j)$, if it is evaluated at α, then $S_k(\alpha) = g_k(\alpha_i; \alpha_j) = a_i$. In other words $\left(f_{i+1}(\alpha_{i+1}; \alpha_{j+1}) = \alpha_{i+1}, f_{i+2}(\alpha_{i+2}; \alpha_{j+2}) = \alpha_{i+2}, \dots f_{i+r}(\alpha_{i+r}; \alpha_{j+r}) = \alpha_{i+r}\right) = (\alpha_{i+1}, \alpha_{i+2}, \dots, \alpha_{i+r})$ which is a local fixed-point of the subnetwork S_k. Therefore, all components of a global fixed-point attractor of F are present in every subnetwork attractor, as long as the rest of the components of α remain constant ■

Definition 14 (Recovering fixed-points from partitioned networks): Let $F: \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean network of *n* nodes. To find the fixed-point attractors of a large network, it can be divided into smaller subnetworks. According to Theorem 7, we know that the attractors of each of these subnetworks contain fragments of the global attractors of the entire network. Therefore, we must combine the attractors of each subnetwork and evaluate them on the original network. If $F(x_a) = x_a$, then the configuration x_a is a true fixed-point attractor of the network F.

Example 12: Consider the Boolean network

$$\begin{cases} x_1^+ = x_2 x_3 \\ x_2^+ = x_1 x_3 \\ x_3^+ = x_2 x_4 \\ x_4^+ = x_1 x_4 \end{cases} \tag{E. 15}$$

For this network, there are only two fixed-points given by $(0,0,0,0)$ and $(1,1,1,1)$. Now, we propose the following subnetworks:

$$S_1 = (x_1^+, x_2^+) = (x_2 x_3, x_1 x_3)$$

$$S_2 = (x_3^+, x_4^+) = (x_2 x_4, x_1 x_4)$$

If we work with S_1 and S_2, we might observe that such subnetworks satisfy $S_1 \cup S_2 = F$ and $S_1 \cap S_2 = \emptyset$. Now, repressing those networks as vectorial functions of defined nodes as follows:

$$g_1(x_1, x_2; x_3) = (x_2 x_3, x_1 x_3)$$

$$g_2(x_1, x_2; x_4) = (x_2 x_4, x_1 x_4)$$

For $x_3 = 0$, we obtain

$$g_1(x_1, x_2; 0) = (0,0)$$

For $x_3 = 1$, we obtain

$$g_1(x_1, x_2; 1) = (x_2, x_1)$$

This function only has two fixed points

$$g_1(x_1, x_2; 1) = (0,0)$$

$$g_1(x_1, x_2; 1) = (1,1)$$

Following the same procedure, the attractors for $g_2(x_1, x_2; 0) = (0,0)$ and $g_2(x_1, x_2; 1) = (0,0), g_2(x_1, x_2; 1) = (1,1)$. Then, all components of the global attractor are parts of the semi-attractors as long as the other components that are not considered in the subnetworks remain constant, as it was stated by Theorem 7.

Now, to recover the global attractors of the full network, according to Definition 9, we need to join the semi-attractors of all vectorial functions, i.e.,

From g_1

$$(0,0,0, x_4)$$

$$(0,0,1,x_4)$$

$$(1,1,1,x_4)$$

From g_2

$$(0,0,x_3,0)$$

$$(0,0,x_3,1)$$

$$(1,1,x_3,1)$$

The evaluation of such semi-attractors at the original network given by $F(x_1,x_2,x_3,x_4) = (x_2x_3, x_1x_3, x_2x_4, x_1x_4)$ produces:

$$F(0,0,0,x_4) = (0,0,0,0)$$

Which implies that x_3, x_4 do not matter for this configuration. Then $F(0,0,0,0) = (0,0,0,0)$ is a global attractor. For the other semi-attractors, we have:

$$F(1,1,1,x_4) = (1,1,x_4,x_4)$$

$$F(1,1,x_3,1) = (x_3,x_3,1,1)$$

Observe that the argument of $F(1,1,1,x_4)$ must be equal to $(1,1,x_4,x_4)$, and the only way to do this, is by making $x_4 = 1$. Thus,

$$F(1,1,1,1) = (1,1,1,1)$$

It confirms that $(1,1,1,1)$ is a global fixed-point, as we already know.

Alternatively, the structural control can be applied in automatic way, as it is shown in the following example.

Example 13: Consider the Boolean network

$$\begin{cases} x_1^+ = x_1 \\ x_2^+ = x_1\bar{x}_3 \\ x_3^+ = x_2 + x_4 \\ x_4^+ = x_4 \end{cases} \quad \text{(E. 16)}$$

Suppose this network allows unlimited structural modifications, and the target configuration is $(1,0,1,1)$. As occurred with the previous example, this network can be divided into two subnetworks given by:

$$g_1(x_1;x_3) = \begin{cases} x_1^+ = x_1 \\ x_2^+ = x_1\bar{x}_3 \end{cases}$$

$$g_2(x_4; x_2) = \begin{cases} x_3^+ = x_2 + x_4 \\ x_4^+ = x_4 \end{cases}$$

Now, if we apply the general form of $\overrightarrow{x^+} = M\vec{\omega}$ for each network, the matrix for these subnetworks is given by:

$$M_1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & \bar{x}_3 & \bar{x}_3 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} x_2 & 1 & x_2 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Observe that each matrix depends on the value of x_2, x_3 to be specified. Considering the control target $(1,0,1,1)$, we might propose the following target matrices:

$$T_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

According to Remark 4, the control laws u_1 and u_2 can be automatically designed as follows:

$$u_1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & \bar{x}_3 & \bar{x}_3 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & \bar{x}_3 & \bar{x}_3 \end{bmatrix}$$

$$u_2 = \begin{bmatrix} x_2 & 1 & x_2 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \oplus \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \bar{x}_2 & 0 & \bar{x}_2 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Next, to determine the algebraic expressions of the control laws, it is necessary to consider the matrix form of each law, for instance:

$$u_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & \bar{x}_3 & \bar{x}_3 \end{bmatrix}$$

For the first row, observe that $u_{1,1}$ is true for the first and second minterms i.e., $u_{1,1} = \bar{x}_1\bar{x}_2 + \bar{x}_1x_2$, which can be simplified as $u_{1,1} = \bar{x}_1$. On the other hand, for the second row, observe that $u_{1,2} = x_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3$, which implies that $u_{1,2} = x_1\bar{x}_3$. Then, the controlled subnetwork g_1 can be expressed as:

$$\begin{cases} x_1^+ = x_1 \oplus u_{1,1} \\ x_2^+ = x_1\bar{x}_3 \oplus u_{1,2} \end{cases}$$

In other words:

$$\begin{cases} x_1^+ = x_1 \oplus \bar{x}_1 \\ x_2^+ = x_1\bar{x}_3 \oplus x_1\bar{x}_3 \end{cases}$$

Applying the same reasoning for u_2, we obtain

$$u_2 = \begin{bmatrix} \bar{x}_2\bar{x}_4 \\ \bar{x}_4 \end{bmatrix}$$

Thus, the controlled subnetwork g_2 is given by:

$$\begin{cases} x_3^+ = (x_2 + x_4) \oplus u_{2,1} \\ x_4^+ = x_4 \oplus u_{2,2} \end{cases}$$

That is:

$$\begin{cases} x_3^+ = (x_2 + x_4) \oplus \bar{x}_2\bar{x}_4 \\ x_4^+ = x_4 \oplus \bar{x}_4 \end{cases}$$

It implies that, the controlled network E.16 is given by:

$$\begin{cases} x_1^+ = x_1 \oplus \bar{x}_1 \\ x_2^+ = x_1\bar{x}_3 \oplus x_1\bar{x}_3 \\ x_3^+ = (x_2 + x_4) \oplus \bar{x}_2\bar{x}_4 \\ x_4^+ = x_4 \oplus \bar{x}_4 \end{cases}$$

It is interesting to note that this controlled network will produce exactly the desired result. However, the structural control aims to be as minimally invasive as possible, since we do not know precisely what the biological constraints of the target system will be. Consequently, it is mandatory to simplify interventions on the network to the minimum. To do this, we need to understand how control signals propagate through the system (i.e., its structural influence). In fact, this is an easy task, since each logical rule depends explicitly on a specific set of logical variables. For instance:

$$x_1^+ = f_1(x_1), \qquad x_2^+ = f_2(x_1, -x_3), \qquad x_3^+ = f_3(x_1, x_4), \qquad x_4^+ = f_4(x_4)$$

Where x_j represents the activation due to the node x_j, and $-x_i$ represents the inhibition due to the node x_i. Now, we need to count how many times a node appears in the logical rules; for instance, x_1 appears in 2 rules. Next, we need to determine the real influence of each node in the logical rules. To do this, the average Boolean sensitivity $I(x_i)$ can be calculated as follows:

$$I(x_k) = \sum_k \Pr[f_r(x) \neq f_r(x \; with \; x_k \; complement)]$$

This formula indicates that it is necessary to evaluate in each logical rule affected by the node x_k what happens if the node x changes its values. For example, for $x_1^+ = x_1$, the sensitivity is 1, because it does not depend of any other nodes except by x_1. In this way, for $x_2^+ = x_1\bar{x}_3$ observe that:

x_3	$x_1 = 0$	$x_1 = 1$	Changes
0	0	1	Yes
1	0	0	No

This means that the influence of x_1 on x_2 is ½ because only when $x_3 = 0$, this node changes the behavior of the node. In consequence, the average Boolean sensitivity for $x_1$is given by:

$$I(x_k) = \sum_k \Pr[f_r(x) \neq f_r(x \; with \; x_k \; complement)] = 1 + 0.5 = 1.5$$

Repeating this procedure for the rest of nodes, we have:

Ranking	Node	Influence	Presence in rules
1	x_1	1.5	2
1	x_4	1.5	2
2	x_2	0.5	1
2	x_3	0.5	1

Thus, fixing x_1 and x_4 can influence the entire network. Then, we need to maintain the control on x_1 and x_4 to obtain the same result of the fully controlled network. In other words:

$$\begin{cases} x_1^+ = x_1 \oplus \bar{x}_1 \\ x_2^+ = x_1 \bar{x}_3 \\ x_3^+ = x_2 + x_4 \\ x_4^+ = x_4 \oplus \bar{x}_4 \end{cases}$$

Which is equivalent to:

$$\begin{cases} x_1^+ = 1 \\ x_2^+ = x_1 \bar{x}_3 \\ x_3^+ = x_2 + x_4 \\ x_4^+ = 1 \end{cases}$$

Therefore, this is the fully controlled network with the minimum of control interventions on its structure.

The most important implication of this section is that it is indeed possible to identify attractors in large Boolean networks, as previously we reported (5). Consequently, the control methods presented here can be applied to regulate fixed points in large networks. While how to efficiently find cyclic attractors for large Boolean networks remains to be determined, the approach shows promise in its current state of development.

Technical details of the CD8+ T lymphocyte network case study

Regulatory T cells expressing FoxP3 have been identified within both $CD4^+$ and $CD8^+$ T-cell compartments (6). $CD4^+$ $FoxP3^+$ regulatory T cells are known to play essential roles in controlling inflammation and regulating the activity of $CD8^+$ T cells (7); however, the study of regulatory phenotypes remains challenging due to their limited stability *in vitro*, where they often lose their regulatory characteristics (6). This raises an intriguing question: could hybrid regulatory-effector phenotypes exist within the $CD8^+$ T-cell lineage? Specifically, is it possible for $CD8^+$ T cells to simultaneously express FoxP3 and T-bet, thereby exhibiting a TcReg/Tc1 hybrid phenotype? We hypothesized that such phenotypes may correspond to synchrony-sensitive attractors and applied the temporal control framework to determine whether specific update schemes could generate these hybrid states.

To implement the time control methodology, we used the properties of the update schemes described in the first sections of this work. Specifically, we used a network we had previously published on CD8+ T lymphocyte differentiation (8), shown below in its notation for implementation in BoolNet, in order to obtain its attractors:

```
targets, factors
TBET, ((IFNy | IL12s & !(IL6s | IL4 | IL10)) & !TBET) & !(IL4 | GATA3 | IL6s | FOXO1)
IFNy, ((IFNy | IFN1 & ((IFNy | TBET | EOMES) & mTORC1 & !(GATA3 | TGFB))) & !IL6s & IL4 |
IL10)
GATA3, ((IL2s & IL4) | EOMES| GATA3) & !(TBET | TGFB | IL6s| IFNy)
IL4, ((IL2s & IL4) | EOMES | GATA3) & !(TBET | TGFB | IL6s | IFNy)
RORyT, (IL6s & TGFB) & !(TBET | FOXP3 | GATA3 | FOXO1)
IL10, ((IL10s | EOMES) & (IL10 & (IFNy | IL6s | TGFB | GATA3))) & mTORC1
FOXP3, ((IL2s & IL12s) & TGFB | FOXP3 | IL4 & FOXO1 & !(IL6s | RORyT))
FOXO1, (ROS | FOXO1) & !(mTORC1 | mTORC2)
EOMES, (ROS | EOMES | IFN1 | FOXO1) & !(mTORC2)
mTORC1, (aa | ROS | IL12 | IL12s | Akt) & !(mTORC2 | PD1 | IL15s)
mTORC2, (GFs | ROS) & !mTORC1
ROS, (Glucose | FFAs | Ceramide | EtOH) & SOD
Akt, IFNy | IL4 | IL10 | mTORC2
GLUT1, Akt & EOMES & !FOXO1
GranzymeB, TBET & !(GATA3 | RORyT | FOXP3 | FOXO1)
SOD, ROS & (FOXO1) & !Ceramide
BCL2, (FOXO1) & !ROS
Casp3, FasL & (ROS | Casp3) & !BCL2
```

The literature shows that some transcription factors act as differentiation activators, committing lymphocyte cell fate toward a specific phenotype, such as IFN-gamma (8). It is also known that the presence of ROS strongly inhibits the differentiation of CD8+ T lymphocytes, promoting their exhaustion and affecting their plasticity (9). Therefore, we wondered what the effect of using specific update functions might be to assess the weight of the presence of these nodes. Furthermore, it was necessary to test the mathematical principles of analysis for this type of network using the composition of update functions. To this end,

we proposed a series of update functions that reflected the following situations: 1) synchronous update (i.e., all nodes are updated) (F_S); 2) the conservation function (no updates) (F_0); 3) an update function that does not affect the state of T-bet, GATA3, mTORC2, ROS, and AKT, as proposed in the asynchronous implementation of the original article where we used the network (8) (F_1). 4) a function that only updates the ROS status (F_2), and finally 5) a function that updates everything except the T-bet status (F_3). These update functions are given by:

$$F_s = \begin{cases} x_1^+ = \big(a_4 + x_1 + x_2 + a_2\overline{(a_3 + x_4 + x_6)}\big)\overline{(a_3 + x_3 + x_4 + x_8)} \\ x_2^+ = \big(a_1 + a_2 + (x_1 + x_2 + x_9)x_{10}\overline{(a_5 + x_3)}\big)\overline{(a_3 + x_4 + x_6)} \\ x_3^+ = (a_6x_4 + x_3 + x_9)\overline{(a_3 + a_5 + x_1 + x_2)} \\ x_4^+ = (a_7 + x_3(a_6 + x_4)\bar{x}_1)\overline{(a_3 + x_2)} \\ x_5^+ = a_3a_5\overline{(x_1 + x_3 + x_7 + x_8)} \\ x_6^+ = \big(a_8 + x_9 + x_6(a_3 + a_5 + x_2 + x_3)\big)x_{10} \\ x_7^+ = (a_2 + a_6)(a_5 + x_4 + x_7 + x_8)\overline{(a_3 + x_5)} \\ x_8^+ = (x_8 + x_{12})\overline{(x_{10} + x_{11})} \\ x_9^+ = (a_4 + x_8 + x_9 + x_{12})\bar{x}_{11} \\ x_{10}^+ = (a_2 + a_6 + a_9 + x_{12} + x_{13})\overline{(a_{10} + a_{11} + x_{11})} \\ x_{11}^+ = (a_{12} + x_{12})\bar{x}_{10} \\ x_{12}^+ = (a_{13} + a_{14} + a_{15} + a_{16})\bar{x}_{16} \\ x_{13}^+ = a_{17} + x_2 + x_4 + x_6 + x_{11} \\ x_{14}^+ = x_9x_{13}\bar{x}_8 \\ x_{15}^+ = x_2\overline{(x_3 + x_5 + x_7 + x_8)} \\ x_{16}^+ = x_8x_{12}\bar{a}_{15} \\ x_{17}^+ = x_8\bar{x}_{12} \\ x_{18}^+ = a_{18}(x_{12} + x_{18})\bar{x}_{17} \end{cases}$$

Where x_1 =T-bet, x_2 =IFN-g, x_3 = GATA3, x_4 = IL-4, x_5 = RORgT, x_6 = IL-10, x_7 = FoxP3, x_8 = FoxO1, x_9 = EOMES, x_{10} = mTORC1, x_{11} = mTORC2, x_{12} = ROS, x_{13} = AKT, x_{14} = GLUT1, x_{15} = Granzyme B, x_{16} = SOD, x_{17} = BCL2, x_{18} = Caspase-3. Here, the inputs are $a_1 = IL12s$, $a_2 = IL6s$, $a_3 = IFN\gamma s$, $a_4 = IFNI$, $a_5 = IL2s$, $a_6 = IL4s$, $a_7 = IL10s$, $a_8 = TGF\beta$, $a_9 = aa$, $a_{10} = GFs$, $a_{11} = PD1$, $a_{12} = IL15s$, $a_{13} = Glucose$, $a_{14} = FFAs$, $a_{15} = a_{16} = Ceramide$, $a_{17} = EtOH$, $a_{18} = FasL$.

$$F_0 = \begin{cases} x_1^+ = x_1 \\ x_2^+ = x_2 \\ x_3^+ = x_3 \\ x_4^+ = x_4 \\ x_5^+ = x_5 \\ x_6^+ = x_6 \\ x_7^+ = x_7 \\ x_8^+ = x_8 \\ x_9^+ = x_9 \\ x_{10}^+ = x_{10} \\ x_{11}^+ = x_{11} \\ x_{12}^+ = x_{12} \\ x_{13}^+ = x_{13} \\ x_{14}^+ = x_{14} \\ x_{15}^+ = x_{15} \\ x_{16}^+ = x_{16} \\ x_{17}^+ = x_{17} \\ x_{18}^+ = x_{18} \end{cases}$$

$$F_1 = \begin{cases} x_1^+ = x_1 \\ x_2^+ = \big(a_1 + a_2 + (x_1 + x_2 + x_9)x_{10}\overline{(a_5 + x_3)}\big)\overline{(a_3 + x_4 + x_6)} \\ x_3^+ = x_3 \\ x_4^+ = (a_7 + x_3(a_6 + x_4)\bar{x}_1)\overline{(a_3 + x_2)} \\ x_5^+ = a_3 a_5 \overline{(x_1 + x_3 + x_7 + x_8)} \\ x_6^+ = \big(a_8 + x_9 + x_6(a_3 + a_5 + x_2 + x_3)\big)x_{10} \\ x_7^+ = (a_2 + a_6)(a_5 + x_4 + x_7 + x_8)\overline{(a_3 + x_5)} \\ x_8^+ = (x_8 + x_{12})\overline{(x_{10} + x_{11})} \\ x_9^+ = (a_4 + x_8 + x_9 + x_{12})\bar{x}_{11} \\ x_{10}^+ = (a_2 + a_6 + a_9 + x_{12} + x_{13})\overline{(a_{10} + a_{11} + x_{11})} \\ x_{11}^+ = x_{11} \\ x_{12}^+ = (a_{13} + a_{14} + a_{15} + a_{16})\bar{x}_{16} \\ x_{13}^+ = x_{13} \\ x_{14}^+ = x_9 x_{13} \bar{x}_8 \\ x_{15}^+ = x_2 \overline{(x_3 + x_5 + x_7 + x_8)} \\ x_{16}^+ = x_8 x_{12} \bar{a}_{15} \\ x_{17}^+ = x_8 \bar{x}_{12} \\ x_{18}^+ = a_{18}(x_{12} + x_{18})\bar{x}_{17} \end{cases}$$

$$F_2 = \begin{cases} x_1^+ = x_1 \\ x_2^+ = x_2 \\ x_3^+ = x_3 \\ x_4^+ = x_4 \\ x_5^+ = x_5 \\ x_6^+ = x_6 \\ x_7^+ = x_7 \\ x_8^+ = x_8 \\ x_9^+ = x_9 \\ x_{10}^+ = x_{10} \\ x_{11}^+ = x_{11} \\ x_{12}^+ = (a_{13} + a_{14} + a_{15} + a_{16})\bar{x}_{16} \\ x_{13}^+ = x_{13} \\ x_{14}^+ = x_{14} \\ x_{15}^+ = x_{15} \\ x_{16}^+ = x_{16} \\ x_{17}^+ = x_{17} \\ x_{18}^+ = x_{18} \end{cases}$$

$$F_3 = \begin{cases} x_1^+ = x_1 \\ x_2^+ = \big(a_1 + a_2 + (x_1 + x_2 + x_9)x_{10}\overline{(a_5 + x_3)}\big)\overline{(a_3 + x_4 + x_6)} \\ x_3^+ = (a_6 x_4 + x_3 + x_9)\overline{(a_3 + a_5 + x_1 + x_2)} \\ x_4^+ = (a_7 + x_3(a_6 + x_4)\bar{x}_1)\overline{(a_3 + x_2)} \\ x_5^+ = a_3 a_5 \overline{(x_1 + x_3 + x_7 + x_8)} \\ x_6^+ = \big(a_8 + x_9 + x_6(a_3 + a_5 + x_2 + x_3)\big)x_{10} \\ x_7^+ = (a_2 + a_6)(a_5 + x_4 + x_7 + x_8)\overline{(a_3 + x_5)} \\ x_8^+ = (x_8 + x_{12})\overline{(x_{10} + x_{11})} \\ x_9^+ = (a_4 + x_8 + x_9 + x_{12})\bar{x}_{11} \\ x_{10}^+ = (a_2 + a_6 + a_9 + x_{12} + x_{13})\overline{(a_{10} + a_{11} + x_{11})} \\ x_{11}^+ = (a_{12} + x_{12})\bar{x}_{10} \\ x_{12}^+ = (a_{13} + a_{14} + a_{15} + a_{16})\bar{x}_{16} \\ x_{13}^+ = a_{17} + x_2 + x_4 + x_6 + x_{11} \\ x_{14}^+ = x_9 x_{13}\bar{x}_8 \\ x_{15}^+ = x_2\overline{(x_3 + x_5 + x_7 + x_8)} \\ x_{16}^+ = x_8 x_{12}\bar{a}_{15} \\ x_{17}^+ = x_8\bar{x}_{12} \\ x_{18}^+ = a_{18}(x_{12} + x_{18})\bar{x}_{17} \end{cases}$$

Next, we proposed a function composition equivalent to the original asynchronous scheme proposed in the article where the network (8) was reported. This function was given by

$$F_T(x) = (F_S \circ F_0 \circ F_0 \circ F_0 \circ F_1 \circ F_0 \circ F_0 \circ F_0)(x)$$

The main text mentioned that this asynchronous update scheme and the synchronous scheme produced the same fixed-point attractors (5). According to Theorem 1 of this work, we know that both schemes share fixed-point attractors. Even so, we used the update functions to compute the fixed-point attractors of both schemes, as described in the main text.

Subsequently, we proposed a new asynchronous update scheme incorporating the update functions F_0, F_2, and F_3. Analyzing these functions, we discovered that they shared seven attractors: four global attractors and three timing-sensitive attractors. After implementing this update scheme, applying F_2 first, then F_0, and finally F_3, we observed the emergence of these timing-sensitive attractors, which were Tc1/Treg hybrids (see Figure 4 in the main text). Our analysis indicates that F_3, containing only seven fixed-point attractors, was the determining factor in guiding the network trajectories toward the emergence of these hybrids. This leads to a very interesting prediction: "transient inhibition of T-bet is critical to ensuring cellular plasticity and the emergence of hybrid CD8+ T lymphocytes." Interestingly, T-bet inhibition has been reported to promote the emergence of several intermediate phenotypes in the terminal differentiation pathway, which could contribute to the emergence of hybrid phenotypes (7). Therefore, through the proposed study, we were able to control network trajectories to generate hybrid phenotypes, which are synchrony-sensitive attractors. This demonstrates the importance of this mathematical theory as a valuable tool for studying the natural control of biological systems, and not just their manipulation.

Technical details of the HSCs network case study

Over the years, the propensity of hematopoietic stem cells to activate and differentiate under normoxic conditions has historically posed a challenge to their accurate study (10). To counteract this effect of ambient oxygen, multiple strategies have been designed to stabilize the quiescent phenotype for longer periods, thus ensuring a reliable reading of their cellular state using flow cytometry (10). In particular, it has been proposed that pharmacological inhibition of the enzymes that degrade the transcription factor HIF-1α, as well as mTOR inhibition with rapamycin, allows for the stabilization of the quiescent phenotype for observation. Regarding yields, it has been reported that stabilizing HIF-1α with Roxadustat maintains between 60% and 75% of the cells in this quiescent state (11), while the use of Rapamycin generates a quiescence rate of between 70% and 85% (10). This background allows us to hypothesize that structural control in the form of pharmacological control can be implemented for this biological system. In consequence, our aim was to find new ways for altering the network dynamics so that the only achievable phenotype is quiescence (i.e., to obtain 100% of undifferentiated quiescent cells).

To implement this, we started by using a previously published network on the differentiation of HSCs (12), which is shown below:

```
targets, factors
Runx1, (PU1 | GATA2 | Runx1) & !Ikzf1
Meis1, (Meis1 | Runx1 | PU1) & !Gfi1
HIF1, (!Oxygen & O2 & !(p53 | FOXO3 | AMPK)) & (!Oxygen & Meis1)
FOXO3, ((AMPK & (HIF1 | p53)) & !(AKT | CEBPA))
p53, HIF1 & !(AKT | GATA1 | PU1)
GATA2, (GATA2 | p53) & !(GATA1 | PU1 | Gfi1)
GATA1, (GATA1 | GATA2 | AKT | Runx1) & !(PU1 | p53 | Ikzf1)
PU1, (PU1 | Runx1 | (CEBPA & Ikzf1)) & !(GATA1 | GATA2 | Gfi1)
CEBPA, (PU1 & Runx1) & !Mef2c
Ikzf1, (Mef2c | Ikzf1 | Runx1) & !(CEBPA | PU1 | GATA1)
Gfi1, (Ikzf1 | CEBPA) & !(PU1 | p53)
Mef2c, PU1 & !CEBPA
mTOR, AKT & (!AMPK | !p53)
AMPK, (!AKT | !OXPHOS & !HIF1) & p53
AKT, H2O2 | !(p53 | FOXO3)
H2O2, ((GATA1 | HIF1 | PU1 | OXPHOS | (SOD & O2)) & !Antiox
O2, !SOD & OXPHOS
SOD, p53 | FOXO3
Antiox, p53 | FOXO3
OXPHOS, (mTOR | AKT) & !(FOXO3 | HIF1)
Oxygen, Oxygen
```

From the network reported in the literature, we know that the attractor corresponding to the HSC has the following characteristics: Meis1, HIF-1a, FOXO3, p53, GATA2, Ikzf1, AMPK, SOD, and ANTIOX are active, while the other nodes remain inactive (12). Therefore, given that it is the desired attractor, we then fragment the network into modules of three nodes each, as follows:

Runx1, (PU1 | GATA2 | Runx1) & !Ikzf1
Meis1, (Meis1 | Runx1 | PU1) & !Gfi1
HIF1, (!Oxygen & O2 & !(p53 | FOXO3 | AMPK)) & (!Oxygen & Meis1)

FOXO3, ((AMPK & (HIF1 | p53)) & !(AKT | CEBPA))
p53, HIF1 & !(AKT | GATA1 | PU1)
GATA2, (GATA2 | p53) & !(GATA1 | PU1 | Gfi1)

GATA1, (GATA1 | GATA2 | AKT | Runx1) & !(PU1 | p53 | Ikzf1)
PU1, (PU1 | Runx1 | (CEBPA & Ikzf1)) & !(GATA1 | GATA2 | Gfi1)
CEBPA, (PU1 & Runx1) & !Mef2c

Ikzf1, (Mef2c | Ikzf1 | Runx1) & !(CEBPA | PU1 | GATA1)
Gfi1, (Ikzf1 | CEBPA) & !(PU1 | p53)
Mef2c, PU1 & !CEBPA

mTOR, AKT & (!AMPK | !p53)
AMPK, (!AKT | !OXPHOS & !HIF1) & p53
AKT, H2O2 | !(p53 | FOXO3)

H2O2, ((GATA1 | HIF1 | PU1 | OXPHOS | (SOD & O2)) & !Antiox
O2, !SOD & OXPHOS
SOD, p53 | FOXO3

Antiox, p53 | FOXO3
OXPHOS, (mTOR | AKT) & !(FOXO3 | HIF1)
Oxygen, Oxygen

Simplifying of the network notation, we obtain:

x1, (x8 | x6 | x1) & !x10
x2, (x2 | x1 | x8) & !x11
x3, (!x21 & x17 & !(x5 | x4 | x14)) & (!x21 & x2)

x4, ((x14 & (x3 | x5)) & !(x15 | x9))
x5, x3 & !(x15 | x7 | x8)
x6, (x6 | x5) & !(x7 | x8 | x11)

x7, (x7 | x6 | x15 | x1) & !(x8 | x5 | x10)
x8, (x8 | x1 | (x9 & x10)) & !(x7 | x6 | x11)
x9, (x8 & x1) & !x12

x10, (x12 | x10 | x1) & !(x9 | x8 | x7)
x11, (x10 | x9) & !(x8 | x5)
x12, x8 & !x9

x13, x15 & (!x14 | !x5)
x14, (!x15 | !x20 & !x3) & x5
x15, x16 | !(x5 | x4)

x16, ((x7 | x3 | x8 | x20 | (x18 & x17)) & !x19)
x17, !x18 & x20
x18, x5 | x4

x19, x5 | x4
x20, (x13 | x15) & !(x4 | x3)
x21, x21

As in Example 13, the network was divided into modules of three nodes, and a suitable control law was sought to guide the dynamics of each subnetwork toward the desired behavior. Regarding the first subnetwork comprised of x_1, x_2, and x_3, we first apply the logical rules algebraically so that only these three nodes are considered variables, while the others are treated as constants. That is:

$$\begin{cases} x_1^+ = (x_8 + x_6 + x_1)\bar{x}_{10} \\ x_2^+ = (x_2 + x_1 + x_8)\bar{x}_{11} \\ x_3^+ = \bar{x}_{21}x_{17}\overline{(x_5 + x_4 + x_{14})}x_2 \end{cases}$$

Now, we defined the logical constants:

$C_1 \coloneqq x_6 + x_8$, $C_2 \coloneqq x_8$, $C_3 \coloneqq \bar{x}_{21}x_{17}\overline{(x_5 + x_4 + x_{14})}x_2$, $A_1 \coloneqq \bar{x}_{10}$ and $A_2 \coloneqq \bar{x}_{11}$.

Substituting in the original subnetwork, we obtain:

$$\begin{cases} x_1^+ = (C_1 + x_1)A_1 \\ x_2^+ = (x_2 + x_1 + C_2)A_2 \\ x_3^+ = C_3x_2 \end{cases}$$

Then, the matrix of the general form of this network $\overrightarrow{x^+} = M_1\vec{\omega}$ is given by:

$$M_1 = \begin{bmatrix} C_1A_1 & C_1A_1 & C_1A_1 & C_1A_1 & A_1 & A_1 & A_1 & A_1 \\ C_2A_2 & C_2A_2 & A_2 & A_2 & A_2 & A_2 & A_2 & A_2 \\ 0 & 0 & C_3 & C_3 & 0 & 0 & C_3 & C_3 \end{bmatrix}$$

According to the control objective, the desired dynamics for this subnetworks is

$$T_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The next step is to find a control law such that $u_1 = M_1 \oplus T_1$, in other words:

$$u_1 = \begin{bmatrix} C_1A_1 & C_1A_1 & C_1A_1 & C_1A_1 & A_1 & A_1 & A_1 & A_1 \\ \overline{C_2A_2} & \overline{C_2A_2} & \overline{A_2} & \overline{A_2} & \overline{A_2} & \overline{A_2} & \overline{A_2} & \overline{A_2} \\ 1 & 1 & \overline{C_3} & \bar{C}_3 & 1 & 1 & \bar{C}_3 & \overline{C_3} \end{bmatrix}$$

Simplifying algebraically, the controlled subnetwork is given by

$$\begin{cases} x_1^+ = (C_1 + x_1)A_1 \oplus (C_1 + x_1)A_1 \\ x_2^+ = (x_2 + x_1 + C_2)A_2 \oplus (\overline{A_2} + \overline{C_2}\overline{x_1x_2}) \\ x_3^+ = C_3x_2 \oplus (\overline{C_3} + \overline{x_2}) \end{cases}$$

Which is equivalent to:

$$\begin{cases} x_1^+ = (x_8 + x_6 + x_1)\bar{x}_{10} \oplus (x_8 + x_6 + x_1)\bar{x}_{10} \\ x_2^+ = (x_2 + x_1 + x_8)\bar{x}_{11} \oplus (x_{11} + \overline{x_1x_2x_8}) \\ x_3^+ = \bar{x}_{21}x_{17}\overline{(x_5 + x_4 + x_{14})}x_2 \oplus \left(\overline{\bar{x}_{21}x_{17}\overline{(x_5 + x_4 + x_{14})}} + \overline{x_2}\right) \end{cases}$$

Applying the same procedure to the rest of the subnetworks and considering the genetic aim, we obtain a controlled network given by

$$\begin{cases} x_1^+ = (x_8 + x_6 + x_1)\bar{x}_{10} \oplus (x_8 + x_6 + x_1)\bar{x}_{10} \\ x_2^+ = (x_2 + x_1 + x_8)\bar{x}_{11} \oplus (x_{11} + \overline{x_1x_2x_8}) \\ x_3^+ = \bar{x}_{21}x_{17}\overline{(x_5 + x_4 + x_{14})}x_2 \oplus \left(\overline{\bar{x}_{21}x_{17}\overline{(x_5 + x_4 + x_{14})}} + \overline{x_2}\right) \\ x_4^+ = x_{14}\,(x_3 + x_5)\overline{(x_9 + x_{15})} \oplus \left(\bar{x}_{21} + (x_9 + x_{15}) + \overline{(x_3 + x_5)}\right) \\ x_5^+ = x_3\overline{(x_7 + x_8 + x_{15})} \oplus \left(\bar{x}_3 + (x_7 + x_8 + x_{15})\right) \\ x_6^+ = (x_5 + x_6)\overline{(x_7 + x_8 + x_{11})} \oplus \left(\overline{(x_5 + x_6)} + (x_7 + x_8 + x_{11})\right) \\ x_7^+ = (x_1 + x_6 + x_7 + x_{15})\overline{(x_5 + x_8 + x_{10})} \oplus (x_1 + x_6 + x_7 + x_{15})\overline{(x_5 + x_8 + x_{10})} \\ x_8^+ = (x_1 + x_8 + x_9x_{10})\overline{(x_6 + x_7 + x_{11})} \oplus (x_1 + x_8 + x_9x_{10})\overline{(x_6 + x_7 + x_{11})} \\ x_9^+ = (x_1 + x_8)\bar{x}_{12} \oplus (x_1 + x_8)\bar{x}_{12} \\ x_{10}^+ = (x_1 + x_{10} + x_{12})\overline{(x_7 + x_8 + x_9)} \oplus \left(\overline{(x_1 + x_{10} + x_{12})} + (x_7 + x_8 + x_9)\right) \\ x_{11}^+ = (x_9 + x_{10})\overline{(x_5 + x_8)} \oplus (x_9 + x_{10})\overline{(x_5 + x_8)} \\ x_{12}^+ = x_8\bar{x}_9 \oplus x_8\bar{x}_9 \\ x_{13}^+ = x_{15}(\bar{x}_5 + \bar{x}_{14}) \oplus x_{15}(\bar{x}_5 + \bar{x}_{14}) \\ x_{14}^+ = x_5(\bar{x}_{15} + \bar{x}_3\bar{x}_{20}) \oplus \left(\bar{x}_5 + \overline{(\bar{x}_{15} + \bar{x}_3\bar{x}_{20})}\right) \\ x_{15}^+ = x_{16}\overline{(x_4 + x_5)} \oplus x_{16}\overline{(x_4 + x_5)} \\ x_{16}^+ = (x_3 + x_7 + x_8 + x_{20} + x_{17}x_{18})\bar{x}_{19} \oplus (x_3 + x_7 + x_8 + x_{20} + x_{17}x_{18})\bar{x}_{19} \\ x_{17}^+ = x_{20}\bar{x}_{18} \oplus x_{20}\bar{x}_{18} \\ x_{18}^+ = (x_4 + x_5) \oplus \overline{(x_4 + x_5)} \\ x_{19}^+ = (x_4 + x_5) \oplus (x_4 + x_5) \\ x_{20}^+ = (x_{13} + x_{15})\overline{(x_3 + x_4)} \oplus (x_{13} + x_{15})\overline{(x_3 + x_4)} \\ x_{21}^+ = x_{21} \end{cases}$$

Observe that, in this controlled network x_{21} has no control law, because the control aim indicates that oxygen is not relevant to maintain the HSC phenotype. After this, it is necessary to reduce the interventions on the network through the analysis of structural influence. To perform such analysis, we need to rewrite logical rules to make explicit their dependences form other nodes. In other words:

$$\begin{cases} x_1^+ = f_1(x_1, x_6, x_8, -x_{10}) \\ x_2^+ = f_2(x_1, x_2, x_8, -x_{11}) \\ x_3^+ = f_3(x_2, -x_4, -x_5, -x_{14}, x_{17}, -x_{21}) \\ x_4^+ = f_4(x_3, x_5, -x_9, x_{14}, -x_{15}) \\ x_5^+ = f_5(x_3, -x_7, -x_8, -x_{15}) \\ x_6^+ = f_6(x_5, x_6, -x_7, -x_8, -x_{11}) \\ x_7^+ = f_7(x_1, -x_5, x_6, x_7, -x_8, -x_{10}, x_{15}) \\ x_8^+ = f_8(x_1, -x_6, -x_7, x_8, x_9, x_{10}, -x_{11}) \\ x_9^+ = f_9(x_1, x_8, -x_{12}) \\ x_{10}^+ = f_{10}(x_1, -x_7, -x_8, -x_9, x_{10}, x_{12}) \\ x_{11}^+ = f_{11}(-x_5, -x_8, x_9, x_{10}) \\ x_{12}^+ = f_{12}(x_8, -x_9) \\ x_{13}^+ = f_{13}(x_5, -x_{14}, -x_{15}) \\ x_{14}^+ = f_{14}(-x_3, x_5, -x_{15}, -x_{20}) \\ x_{15}^+ = f_{15}(-x_4, -x_5, x_{16}) \\ x_{16}^+ = f_{16}(x_3, x_7, x_8, x_{17}, x_{18}, -x_{19}, x_{20}) \\ x_{17}^+ = f_{17}(-x_{18}, x_{20}) \\ x_{18}^+ = f_{18}(x_4, x_5) \\ x_{19}^+ = f_{19}(x_4, x_5) \\ x_{20}^+ = f_{20}(-x_3, -x_4, x_{13}x_{15}) \\ x_{21}^+ = f_{21}(x_{21}) \end{cases}$$

From this, we can count how many times a node appears in the logical rules; for example, x_8 appears in 11 rules, x_5 appears in 10 rules, and x_4 appears in 5 rules. However, simply appearing many times in the logical rules is not enough; what really matters is the influence of changes in a node on the logical rules. To determine this, the average Boolean sensitivity $I(x_i)$ was calculated. For instance, for $x_{18}^+ = x_4 + x_5$, the following was obtained:

x_4	$x_5 = 0$	$x_5 = 1$	Changes
0	0	1	Yes
1	1	1	No

Then, the influence of x_5 on x_{18} is ½ = 0.5. This procedure was repeated at all nodes directly influenced by x_5, that is: $x_3, x_4, x_6, x_7, x_{11}, x_{13}, x_{14}, x_{15}, x_{18}$ and x_{19}. As a result of this, the average Boolean sensitivity for x_5, was $I(x_5) \approx 2.89$. Consequently, by repeating this process for all nodes in the network, it was found that:

Ranking	Node	Influence	Presence in rules
1	x_5 $(p53)$	2.89	10
2	x_8 $(PU1)$	2.23	11
3	x_4 $(FOXO3)$	1.66	5
4	$x_{15}(AKT)$	1.58	6
5	$x_{10}(Ikzf1)$	1.28	5
6	$x_{11}(Gfi1)$	1.27	3
7	x_9 $(CEBPA)$	1.05	5
8	x_{21} $(Oxygen)$	1.03	2

As can be seen, x_5 (p53) is a node that must be on in the control objective. However, if we look at the relationship between x_5 and x_3, we will notice that $x_5 = 1$ suppresses x_3. Consequently, we must leave x_3 on to maintain the control objective. Furthermore, x_4 (FOXO3) must also be activated, and its presence will ensure the activation of AMPK, SOD, and Antiox. However, these interventions cannot activate x_2 (Meis1) or x_{10} (Ikzf1). Therefore, it is necessary to maintain control over these two genes to guarantee their activation. Thus, the controlled network is given by:

$$
\begin{cases}
x_1^+ = (x_8 + x_6 + x_1)\bar{x}_{10} \\
x_2^+ = (x_2 + x_1 + x_8)\bar{x}_{11} \oplus (x_{11} + \overline{x_1 x_2 x_8}) \\
x_3^+ = \bar{x}_{21} x_{17} \overline{(x_5 + x_4 + x_{14})} x_2 \oplus \left(\overline{\bar{x}_{21} x_{17} \overline{(x_5 + x_4 + x_{14})}} + \overline{x_2} \right) \\
x_4^+ = x_{14}\,(x_3 + x_5)\overline{(x_9 + x_{15})} \oplus \left(\bar{x}_{21} + (x_9 + x_{15}) + \overline{(x_3 + x_5)} \right) \\
x_5^+ = x_3 \overline{(x_7 + x_8 + x_{15})} \oplus \left(\bar{x}_3 + (x_7 + x_8 + x_{15}) \right) \\
x_6^+ = (x_5 + x_6)\overline{(x_7 + x_8 + x_{11})} \\
x_7^+ = (x_1 + x_6 + x_7 + x_{15})\overline{(x_5 + x_8 + x_{10})} \\
x_8^+ = (x_1 + x_8 + x_9 x_{10})\overline{(x_6 + x_7 + x_{11})} \\
x_9^+ = (x_1 + x_8)\bar{x}_{12} \\
x_{10}^+ = (x_1 + x_{10} + x_{12})\overline{(x_7 + x_8 + x_9)} \oplus \left(\overline{(x_1 + x_{10} + x_{12})} + (x_7 + x_8 + x_9) \right) \\
x_{11}^+ = (x_9 + x_{10})\overline{(x_5 + x_8)} \\
x_{12}^+ = x_8 \bar{x}_9 \\
x_{13}^+ = x_{15}(\bar{x}_5 + \bar{x}_{14}) \\
x_{14}^+ = x_5(\bar{x}_{15} + \bar{x}_3 \bar{x}_{20}) \\
x_{15}^+ = x_{16}\overline{(x_4 + x_5)} \\
x_{16}^+ = (x_3 + x_7 + x_8 + x_{20} + x_{17} x_{18})\bar{x}_{19} \\
x_{17}^+ = x_{20}\bar{x}_{18} \\
x_{18}^+ = (x_4 + x_5) \\
x_{19}^+ = (x_4 + x_5) \\
x_{20}^+ = (x_{13} + x_{15})\overline{(x_3 + x_4)} \\
x_{21}^+ = x_{21}
\end{cases}
$$

Which is equivalent to:

$$\begin{cases} x_1^+ = (x_8 + x_6 + x_1)\bar{x}_{10} \\ x_2^+ = 1 \\ x_3^+ = 1 \\ x_4^+ = 1 \\ x_5^+ = 1 \\ x_6^+ = (x_5 + x_6)\overline{(x_7 + x_8 + x_{11})} \\ x_7^+ = (x_1 + x_6 + x_7 + x_{15})\overline{(x_5 + x_8 + x_{10})} \\ x_8^+ = (x_1 + x_8 + x_9 x_{10})\overline{(x_6 + x_7 + x_{11})} \\ x_9^+ = (x_1 + x_8)\bar{x}_{12} \\ x_{10}^+ = 1 \\ x_{11}^+ = (x_9 + x_{10})\overline{(x_5 + x_8)} \\ x_{12}^+ = x_8\bar{x}_9 \\ x_{13}^+ = x_{15}(\bar{x}_5 + \bar{x}_{14}) \\ x_{14}^+ = x_5(\bar{x}_{15} + \bar{x}_3\bar{x}_{20}) \\ x_{15}^+ = x_{16}\overline{(x_4 + x_5)} \\ x_{16}^+ = (x_3 + x_7 + x_8 + x_{20} + x_{17}x_{18})\bar{x}_{19} \\ x_{17}^+ = x_{20}\bar{x}_{18} \\ x_{18}^+ = (x_4 + x_5) \\ x_{19}^+ = (x_4 + x_5) \\ x_{20}^+ = (x_{13} + x_{15})\overline{(x_3 + x_4)} \\ x_{21}^+ = x_{21} \end{cases}$$

Therefore, this control method indicates that, for 100% of the configurations to achieve the HSC phenotype, it is necessary to constitutively activate Meis1, HIF-1, FOXO3, p53 y Ikzf1, which is congruent with previous experimental observations. It is interesting to note that this control method predicts that to increase the conversion yield of the HSC phenotype it is necessary to stimulate the expression of more genes than simply stabilizing HIF-1 as is done in the literature (11).

Supplementary references

1. Bensussen, A., Díaz, J. (2015). Dynamics of p53 and Cancer, Evolution of Ionizing Radiation Research, Mitsuru Nenoi, IntechOpen, DOI: 10.5772/60916.
2. Victor J, Deutsch J, Whitaker A, Lamkin EN, March A, Zhou P, Botten JW, Chatterjee N. (2021). SARS-CoV-2 triggers DNA damage response in Vero E6 cells. Biochem Biophys Res Commun.579:141-145. doi: 10.1016/j.bbrc.2021.09.024.
3. Zauli G, AlHilali S, Al-Swailem S, Secchiero P, Voltan R. (2022). Therapeutic potential of the MDM2 inhibitor Nutlin-3 in counteracting SARS-CoV-2 infection of the eye through p53 activation. Front Med (Lausanne). 9:902713. doi: 10.3389/fmed.2022.902713.
4. Chaves M, Tournier L. (2018). Analysis Tools for Interconnected Boolean Networks With Biological Applications. Front Physiol. 9:586. https://doi.org/10.3389/fphys.2018.00586
5. Bensussen A, Arciniega-González JA, Álvarez-Buylla ER, Martínez-García JC. (2025). Analytical approach of synchronous and asynchronous update schemes applied to solving biological Boolean networks. PLoS One 20(9): e0319240. https://doi.org/10.1371/journal.pone.0319240
6. Bolivar-Wagers, S., Larson, J. H., Jin, S. & Blazar, B. R. (2022). Cytolytic CD4+ and CD8+ Regulatory T-Cells and Implications for Developing Immunotherapies to Combat Graft-Versus-Host Disease. Front. Immunol. 13, 864748.
7. Mambetsariev, N., Torres Acosta, M. A., Lee, S., Tang, A., Bell, B. L., Chacon-Solano, E., Campbell, D. J. & Delacher, M. (2025). FOXP3+ Regulatory T Cells Require TBET to Regulate Activated CD8+ T Cells During Recovery From Influenza. Am. J. Respir. Cell Mol. Biol. 72, 453–456.
8. Bensussen, A. Santana, M. A. Rodríguez-Jorge, O. (2022) Metabolic Alterations Impair Differentiation and Effector Functions of CD8+ T Cells. Frontiers in Immunology (August), pp. 1 - 13. doi: 10.3389/fimmu.2022.945980
9. Peng HY, Lucavs J, Ballard D, Das JK, Kumar A, Wang L, Ren Y, Xiong X, Song J. (2021). Metabolic Reprogramming and Reactive Oxygen Species in T Cell Immunity. Front Immunol. 12:652687. doi: 10.3389/fimmu.2021.652687.
10. Kobayashi H, Takubo K. (2020). Protocol for the Maintenance of Quiescent Murine Hematopoietic Stem Cells. STAR Protoc. Aug 2;1(2):100078. doi: 10.1016/j.xpro.2020.100078.
11. Zhang J, Yao M, Xia S, Zeng F, Liu Q. (2025). Systematic and comprehensive insights into HIF-1 stabilization under normoxic conditions: implications for cellular adaptation and therapeutic strategies in cancer. Cell Mol Biol Lett. 30(1):2. doi: 10.1186/s11658-024-00682-7.
12. Herrera, J. Bensussen, A., García-Gómez, M., Garay-Arroyo, A., Álvarez-Buylla, E.R. (2024). A system-level model reveals that Transcriptional Stochasticity is required for Hematopoietic Stem Cell Differentiation. npj Syst Biol Appl 10, 145.